PVRShaman

User Manual

Copyright © Imagination Technologies Limited. All Rights Reserved.

Appendix A. This publication contains proprietary information which is subject to change without notice and is supplied 'as is' without warranty of any kind. Imagination Technologies and the Imagination Technologies logo are trademarks or registered trademarks of Imagination Technologies Limited. All other logos, products, trademarks and registered trademarks are the property of their respective owners.

Filename	:	PVRShaman.User Manual
Version	:	PowerVR SDK REL_3.5@3523383a External Issue
Issue Date	:	17 Apr 2015
Author	:	Imagination Technologies Limited

Contents

1. Introduction 5
1.1. Document Overview 5
1.2. Software Overview 5
1.2.1. Features 5
1.2.2. Limitations 5
1.2.3. File Types 6
1.3. Installation 6
1.3.1. Installing from Installer 6
1.3.2. Compatibility 6
1.3.3. Optional Requirements 6
2. PVRShaman GUI 7
2.1. User Interface Layout 7
2.1.1. Effects Editor 7
2.1.2. Task List 8
2.2. Menu Bar 9
2.2.1. File Menu 9
2.2.2. Edit Scene Menu 11
2.2.3. Edit Shader Menu 13
2.2.4. Build Menu 14
2.2.5. Render Menu 15
2.2.6. View Menu 16
2.2.7. Tools Menu 16
2.2.8. Help Menu 17
2.3. Working with a Scene 17
2.3.1. Visualizing a Scene 17
2.3.2. Exploring a Scene 18
2.3.3. Adding Materials to a Scene 20
2.3.4. Modifying Properties 23
2.4. Building an Effect 27
2.4.1. PFX Blocks 27
2.4.2. Semantics 30
2.4.3. PVRShaderEditor 30
2.4.4. Profile Output Panel 31
2.5. Render-to-Texture and Post-Process Effects 32
2.5.1. Overview 32
2.5.2. Render-to-Texture 33
2.5.3. Post-Process Effects 34
2.6. Render Modes 36
2.6.1. Effects 36
2.6.2. No Effects 36
2.6.3. Wireframe 37
2.6.4. Wireframe No Effects 37
2.6.5. Depth Complexity 38
2.7. Texture Viewer 38
3. Preferences 39
3.1. General 39
3.2. Viewport 40
3.3. Compiler Settings 41
3.4. Editor Settings 42
3.5. Editor Colours 43
4. Contact Details 44
Appendix B. PVRShaman PFX Semantics List 45
B.1. Attributes 45
B.2. Uniforms 45

Appendix C. Regular Expression Syntax....48

List of Figures

Figure 1. PVRShaman GUI....7
Figure 2. Effects Editor panel....8
Figure 3. Task List window....8
Figure 4. File menu....9
Figure 5. POD Info dialog....10
Figure 6. Edit Scene menu....11
Figure 7. Edit Shader menu....13
Figure 8. Find and Replace dialog....14
Figure 9. Build menu....14
Figure 10. Render menu....15
Figure 11. View menu....16
Figure 12. Tools menu....16
Figure 13. Help menu....17
Figure 14. Visualization Panel....18
Figure 15. Scene Explorer window....19
Figure 16. POD Explorer window....19
Figure 17. Object Data dialog....20
Figure 18. Add Material dialog....21
Figure 19. Creating a blank effects file....22
Figure 20. Light Properties dialog....24
Figure 21. Camera Properties dialog....25
Figure 22. Material Properties dialog....26
Figure 23. Per-line cycle counts....31
Figure 24. Profile Output Panel....31
Figure 25. Post-process effects example....35
Figure 26. Effects render mode....36
Figure 27. No Effects render mode....36
Figure 28. Wireframe render mode....37
Figure 29. Wireframe No Effects render mode....37
Figure 30. Depth Complexity render mode....38
Figure 31. Texture Viewer dialog....38
Figure 32. General tab....39
Figure 33. Viewport tab....40
Figure 34. Compiler Settings tab....41
Figure 35. Editor Settings tab....42
Figure 36. Editor Colours tab....43

List of Tables

Table 1. File types....6
Table 2. Library effects....23
Table 3. PFX header keywords....27
Table 4. Shader block keywords....28

Table 5. Texture block keywords 28
Table 6. Target block keywords 29
Table 7. Effect block keywords 30
Table 8. Profile Output Panel 32
Table 9. General options 39
Table 10. Viewport options 40
Table 11. Compiler Settings options 41
Table 12. Editor Settings options 42
Table 13. Editor Colours options 43
Table 14. Attribute list 45
Table 15. Uniform list 45
Table 16. Regular expression syntax and description 48

1. Introduction

1.1. Document Overview

The purpose of this document is to serve as a complete user manual for the PVRShaman Shader Development Environment. It includes compatibility information, installation instructions, a guide to the functionality of the application and a complete listing of all interface options and preferences.

1.2. Software Overview

PVRShaman is a tool for the rapid prototyping, development, and testing of OpenGL ES 2.0, OpenGL, DirectX9 and DirectX10 shaders. It provides a universal shader compiler and editor with syntax highlighting and per-line cycle counts for rapid shader development and a `Visualization Panel` where the results of shader changes can be seen in real time.

Shader development is done in either PowerVR Effects (PFX) or Microsoft Effect (FX) format, while geometry is handled in the PowerVR Object Data (POD) format; a file format created using the Collada2POD or PVRGeoPOD file exporters also available from the PowerVR Graphics SDK installer.

1.2.1. Features

PVRShaman has a variety of notable features:

- Windows, Linux and OS X support.
- Geometry input from POD files generated using PVRGeoPOD.
- Support for PFX.
- Texture input from PVR files generated using PVRTexTool.
- Support for all common texture formats.
- Integrated version of PVRShaderEditor.
- POD viewer.
- WYSIWYG concept allows rapid prototyping of new shaders.

1.2.2. Limitations

Due to ongoing development, the current edition of PVRShaman has the following limitations:

- Cannot open multiple POD files simultaneously.
- Cannot merge POD files.
- Does not allow for extra lights, cameras or objects to be added to a POD, only materials.
- The default shader (i.e., the shader applied to objects when no shader has been set by the user) only supports one light. For more lights the user must use a shader that supports multiple lights.

1.2.3. File Types

Through the course of this document several file types are mentioned. The most important of these are listed in Table 1.

Table 1. File types

File type	Description
PFX	PFX is a file format for storing and setting up shaders and effects with runtime available in the PowerVR Graphics SDK tools and examples in the training course demos. PFX files are editable in PVRShaman and PVRShaderEditor.
POD	POD is a file format for storing complete scenes with meshes, lights, animations, materials and references to textures and effects. While PODs are not compressed, they store information in a format designed for speedy deployment to hardware.
PVR	PVR is a file format for storing API-friendly textures. PVR files are produced using PVRTexTool and can be compressed as small as 2/4 bits per pixel, while still maintaining a good quality.
Collada (DAE)	Collada is a royalty-free XML file scheme developed by the Khronos group that acts as an interchange format for digital assets and effects.
Microsoft Effects (FX)	Microsoft Effects is a proprietary file format belonging to the Microsoft Corporation used for setting up effects for use with the DirectX API.

1.3. Installation

In order to use PVRShaman as a shader development environment, a graphics card that supports the targeted API is required, e.g., if shaders are being written for DirectX10, a DirectX10 capable graphics card is required. For viewing POD files, an older card can be used.

1.3.1. Installing from Installer

Download the PowerVR Graphics SDK package and follow the on-screen instructions. Once the package is installed browse to:

```
<InstallDir>\PVRShaman\
```

1.3.2. Compatibility

For PVRShaman to run efficiently, it is recommended to install the utility on a machine running at least either Windows 7, OS X 10.6 (Snow Leopard), or Linux Ubuntu 12.04.

1.3.3. Optional Requirements

It is possible to plug PVRTexTool into PVRShaman, allowing for images to be directly opened from the `Scene Explorer` by double-clicking. For this to work, PVRTexTool must be installed and its location set in `Path to PVRTexTool` under `Tools -> Preferences -> General`.

2. PVRShaman GUI

2.1. User Interface Layout

The PVRShaman interface is composed of several panels which hold information about the current scene, the materials within the scene, and any debug output from those materials. The information is presented as portrayed in Figure 1.

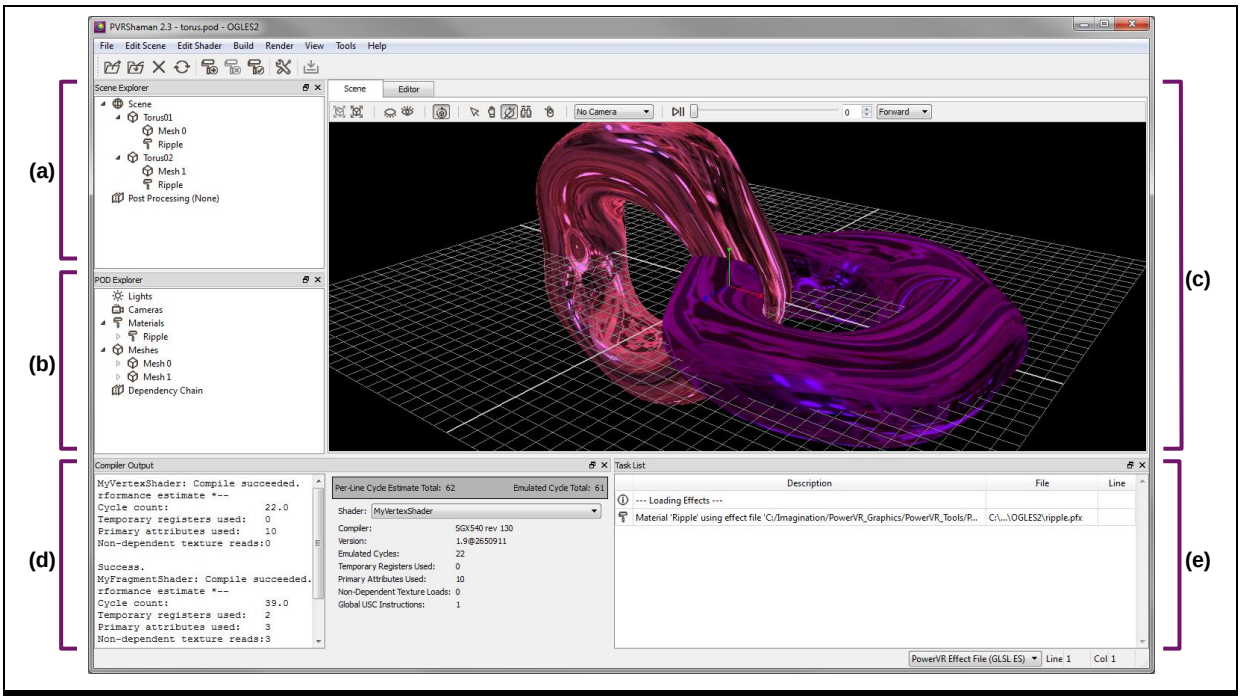

Figure 1. PVRShaman GUI

At the left hand side of the GUI are the windows for the `Scene Explorer` (Figure 1a) and `POD Explorer` (Figure 1b). These are detailed further in Section 2.3.1. The majority of the GUI is taken up by the `Visualization Panel` (Figure 1c), which displays POD files, or can be switched to display the Shader Editor with compiler output information displayed in the `Compiler Output` window (Figure 1d). Figure 1e displays the `Task List` window, which details the state of the loaded files and any error that they may contain.

2.1.1. Effects Editor

The `Effects Editor` panel and the `Compiler Output` window are the home of PVRShaderEditor (see Section 2.4.3). PVRShaderEditor can be used to edit GLSL code, PFX and FX files as well as plain text files. In addition to the functionality already described, opened files are arranged as tabs, so that multiple files can be opened at any given time (Figure 2). Right-clicking within an open file brings up a context menu that allows for easy insertion of attributes or uniforms as well as the normal array of copy and paste options.

Note: Full information on the functionality found in PVRShaderEditor can be accessed in the "PVRShaderEditor User Manual".

Figure 2. Effects Editor panel

2.1.2. Task List

The `Task List` window, used as a debug output panel (Figure 1d), contains information as to the state of the loaded effects and meshes, as displayed in Figure 3. In the case of errors in shaders or effects files, the line number and a description of the error will be seen. Warnings are also given for errors which are 'non-fatal' and should not affect the running of the shader or effect in which the error occurs.

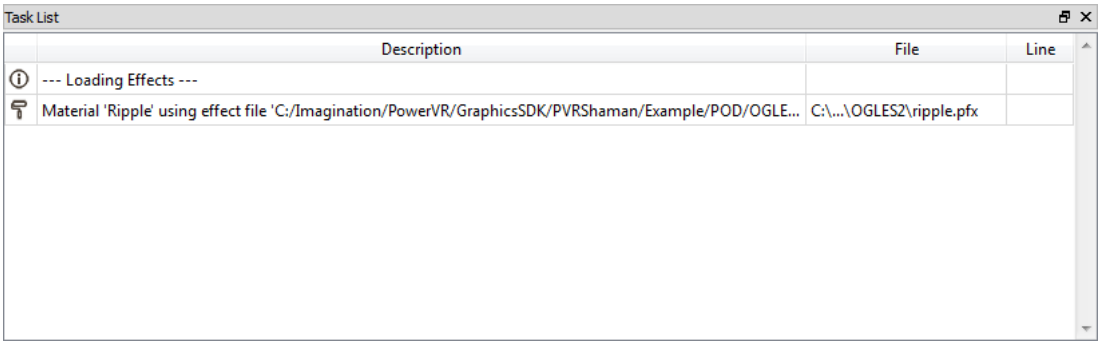

Figure 3. Task List window

2.2. Menu Bar

2.2.1. File Menu

The `File` menu (Figure 4) is used for saving or opening files used by PVRShaman.

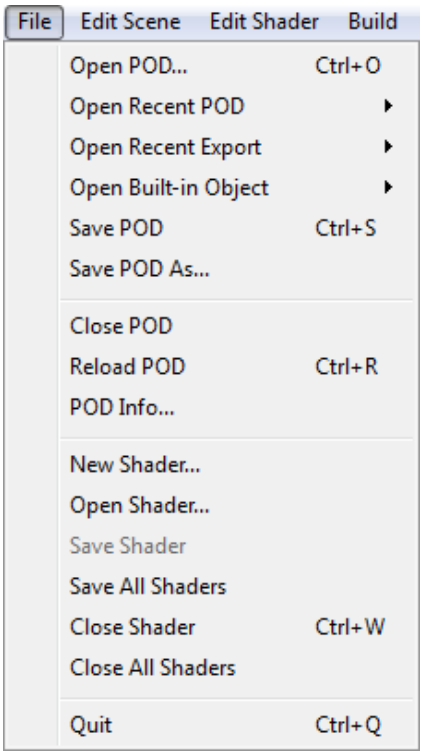

Figure 4. File menu

Open a POD File

`Open POD…` opens a window displaying the location of saved POD files, through which it is possible to select a particular file to open. It is also possible to open files through the toolbar shortcut.

Open a Recent POD File

To continue working on a recently opened POD, selecting `Open Recent POD` will display a list of the ten most recently opened files to choose from. Selecting `Open Recent Export` will display a list of the ten most recently exported files.

Clear Recent Files

To clear the list of the ten most recently opened files, select `Clear recent files` from the relevant option – either `Open Recent POD` or `Open Recent Export`.

Open a Built-in Object

`Open Built-in Object` creates a new POD file from the existing object library.

Save a POD File

`Save POD` saves the currently open POD file. The main toolbar also has a shortcut for this function. If it is required to save the currently open POD file to a new file, select `Save POD As…` and type in a new name for the file.

Close a POD File

`Close POD` closes the currently open POD file, which can also be achieved by clicking the shortcut on the toolbar.

Reload a POD File

It is possible to reload the currently opened POD file by either selecting `Reload POD` from the menu, or through the toolbar. This is particularly useful if the file has been updated from an external application, for example Autodesk 3ds Max.

Display POD File Information

`POD Info` displays information as to the contents of the currently loaded POD file, as shown in Figure 5. The total combined information about the file is displayed at the top with per-mesh information at the bottom (including what information was exported by Collada2POD or PVRGeoPOD). The history section can also be used to keep track of changes made to the file.

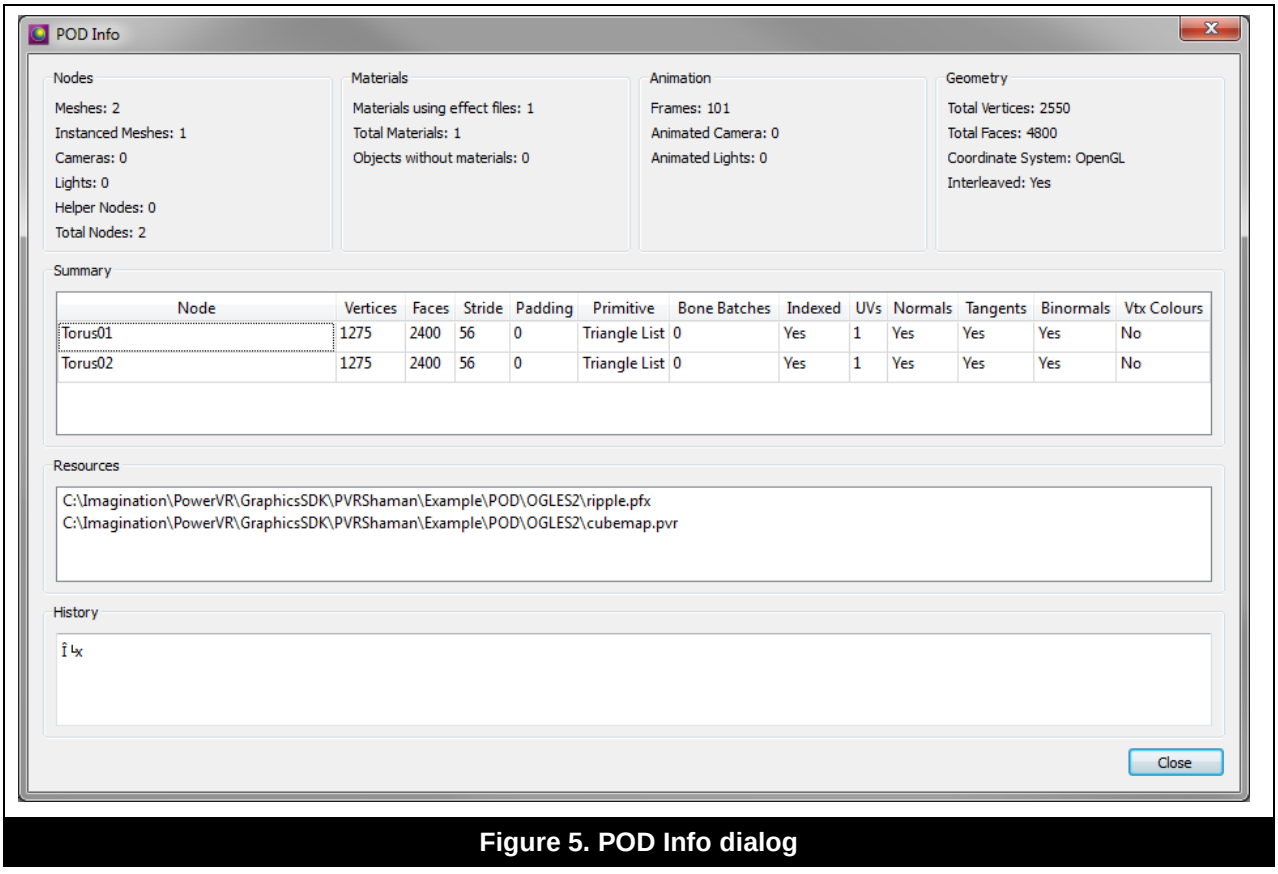

Figure 5. POD Info dialog

Create a New Shader File

The `New Shader` menu contains a list of shader types which, when selected, create a new file of the chosen type.

Open a Shader File

`Open Shader` opens an existing shader file. This can also be achieved by clicking the `Open` icon in the Shader Editor toolbar.

Save Shader Files

`Save Shader` saves the currently selected shader, whereas `Save All Shaders` saves all the currently open shaders. The `Shader Editor` toolbar features a shortcut for the `Save Shader` action.

Close Shader Files

`Close Shader` closes the currently selected shader, whilst `Close All Shaders` closes all open shader files. Shader files can also be closed via the Shader Editor toolbar.

Exit the GUI

`Quit` closes PVRShaman.

2.2.2. Edit Scene Menu

The `Edit Scene` menu (Figure 6) is used when altering the appearance of the scene being displayed in the `Visualization Panel`.

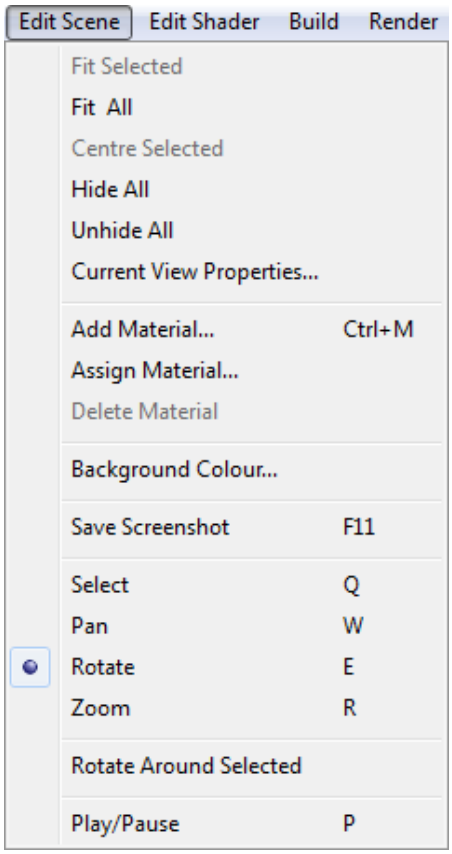

Figure 6. Edit Scene menu

Fit Objects

`Fit Selected` will zoom the `Visualization Panel` in/out to fit the currently selected object, and centres the object within the panel. Selecting `Fit All` will instead zoom the Visualization Panel in/out to fit the entire scene, and centres the scene within the panel. These functions can also be triggered via the panel toolbar.

Centre Selected

To simply centre the current view in the panel on the currently selected object, click `Centre Selected`.

Toggle between Showing and Hiding Objects

`Hide All` hides all objects in the scene, and `Unhide All` undoes the action. Both of these functions can be triggered from the toolbar.

Modify Camera Properties

`Current View Properties` displays the `Modify Camera Properties` dialog for the currently selected camera.

Add Material

`Add Material` brings up the `Add Material` dialog as detailed in the Add Material section.

Assign Material

`Assign Material` is used to assign a given material to a given mesh.

Delete Material

To delete a selected material, click `Delete Material`, or select the associated icon from the main toolbar.

Change the Background Colour

`Background Colour…` changes the background colour of the `Visualization Panel`.

Save a Screenshot

`Save Screenshot` saves a screenshot of the current contents of the `Visualization Panel`.

Select an Object

This radio button changes the control scheme of the `Visualization Panel` so that objects can be selected by clicking them. This option can also be selected via the toolbar.

Pan

This radio button changes the control scheme of the `Visualization Panel` so that it is possible to pan around the scene using the mouse while the left mouse button is down. The control scheme can also be changed by clicking the relevant button in the toolbar. It is also possible to hold the `Shift` key while middle-clicking and dragging with the mouse in order to pan.

Rotate

With `Rotate` active (either through the menu or the toolbar) it is possible to rotate the scene around the currently selected object, the world axis, or the centre of the `Visualization Panel` using the mouse while the left mouse button is down. Rotation can also achieved by middle-clicking and dragging with the mouse.

Zoom in and out

With `Zoom` active, mouse movement will control the zoom level of the `Visualization Panel` while the left mouse button is held down. Pulling back will zoom out, and pushing forwards will zoom in. This function can also be activated from the toolbar. The mouse wheel can also be used to zoom in and out.

FPS Navigation

With this option selected the mouse and keyboard now allow for First Person Shooter (FPS) like movement. Pressing the `Esc` key will exit this mode, returning the control scheme to `Select`.

Rotate View around Selected Object

When this option is ticked, the `Rotate` button on the toolbar rotates around the selected object rather than around the origin of the POD scene.

Play and Pause Animation

`Play/Pause` is used to control the animation currently playing in the `Visualization Panel`. These functions are also available on the panel toolbar.

2.2.3. Edit Shader Menu

The `Edit Shader` menu (Figure 7) is used to modify any open shader files.

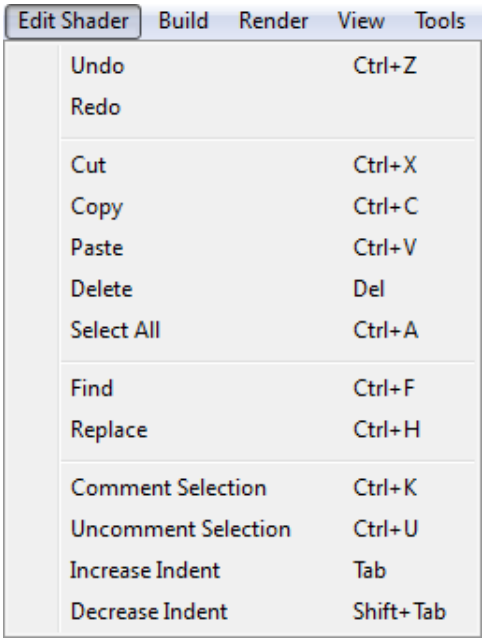

Figure 7. Edit Shader menu

Undo an Action

`Undo` undoes the last performed action.

Redo an Action

The last performed action can be redone by selecting `Redo`. You can also select this function via the `Shader Editor toolbar`.

Cut Text

`Cut` cuts the selected text to the clipboard.

Paste Text

To retrieve the text from the clipboard, select `Paste`.

Delete Text

If it is simply required to delete the currently selected text, click `Delete`.

Select All Text

`Select All` selects the entire contents of the file that currently has focus.

Find and Replace Text

`Find` or `Replace` opens the `Find/Replace dialog` (Figure 8). It can also be accessed via the `Effects Editor toolbar`, or by pressing `Ctrl+F` (to find) or `Ctrl+H` (to replace). Only the relevant sections of the toolbar will be displayed, and any visible sections will appear below the toolbar.

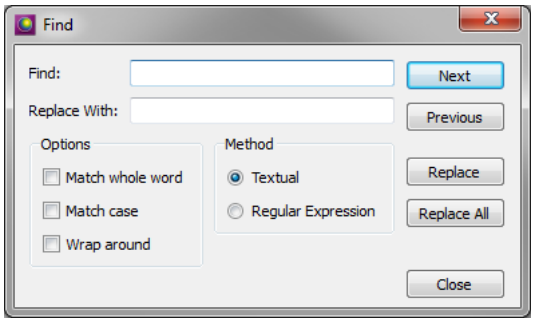

Figure 8. Find and Replace dialog

When ticking the `Advanced` box and set `Method:` to be `Regular Expression` the find bar will interpret the search term as a regular expression. The `Find` field will be shown in red if the search term is not a complete regular expression. Information on what regular expressions are supported can be found in Appendix C. Back references (`\1` to `\9`) can also be used in the `Replace With` field as well as the `Find` field.

Comment on Selected Text

`Comment Selection` comments out the selected text from the file. If a complete line is selected that line will begin with `//`. If part of a line is selected that part will be surrounded in `/* */`.

Uncomment Selected Text

`Uncomment Selection` removes the commenting out from the selected text, either removing a surrounding `/* */` or the `//` at the beginning of the line.

Increase Text Indent

`Increase Indent` indents the selected text by a single tab.

Decrease Text Indent

`Decrease Indent` removes an indent from the selected text.

2.2.4. Build Menu

The `Build` menu (Figure 9) is used for applying and rebuilding shaders.

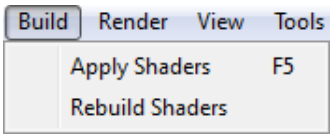

Figure 9. Build menu

Apply Shaders

`Apply Shaders` compiles all shaders in the current POD file that have been edited and applies them to the scene in the `Visualization Panel`. This function can also be triggered via a shortcut on the toolbar.

Rebuild Shaders

`Rebuild Shaders` rebuilds all shaders in the current POD file irrespective of whether that shader has been edited.

2.2.5. Render Menu

The `Render` menu (Figure 10) is used when dealing with the meshes of objects in a scene.

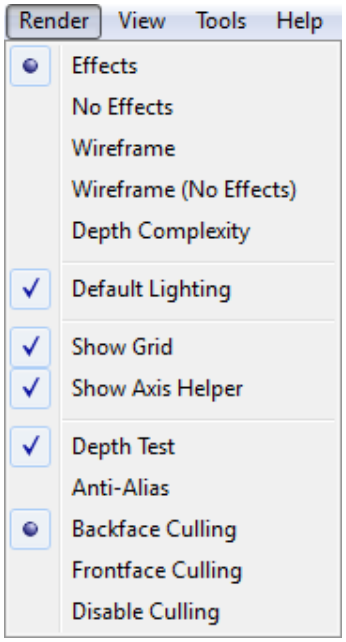

Figure 10. Render menu

Render Modes

The first five options in the `Render` menu represent the available render modes:

- Effects.
- Wireframe.
- Wireframe No Effects.
- No Effects.
- Depth Complexity.

More details on these render modes can be found in Section 2.6 of this manual.

Toggle Default Lighting on and off

`Default Lighting` toggles the default light on or off. When it is turned off, only lights from the POD file are used.

Show or Hide Grid

`Show Grid` displays a grid around the origin within the `Visualization Panel`.

Show or Hide Axis Helper

`Show Axis Helper` displays a small X, Y, Z axis diagram around the origin within the Visualization Panel.

Enable or Disable Depth Test

`Depth Test` allows for the enabling or disabling of depth testing within the scene.

Activate or Deactivate Anti-Aliasing

`Anti-Alias` activates or deactivates anti-aliasing for the current scene. Toggling this option requires the scene to be saved and reopened.

Enable or Disable Culling

`Backface Culling` tells PVRShaman to enable or disable back-face culling. `Frontface Culling` enables or disables front-face culling. This allows scenes that have different winding orders for their polygons to be rendered correctly by PVRShaman. `Disable Culling` disables all culling.

2.2.6. View Menu

The `View` menu (Figure 11) is used to access various display panels in the GUI.

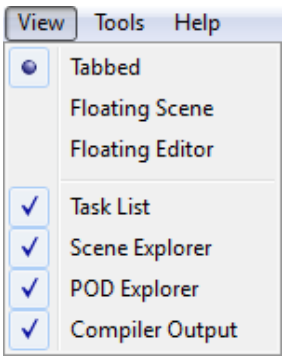

Figure 11. View menu

Customize the GUI Layout

The `Visualization Panel` and the `Effects/Shader Editor` are displayed as tabs by default. By selecting either `Floating Scene` or `Floating Editor`, that panel can be opened up as its own window for ease of viewing.

Show or Hide the Task List Window

Selecting the `Task List` option will display the relevant section of the GUI.

Show or Hide the Scene Explorer Window

To display the `Scene Explorer` in the GUI, select the relevant option from the menu.

Show or Hide the POD Explorer Window

To display the `POD Explorer` in the GUI, select the relevant option from the menu.

Show or Hide the Compiler Output Window

If the `Compiler Output` option is ticked, then the relevant panel will be visible on the GUI.

2.2.7. Tools Menu

The `Tools` menu (Figure 12) allows accessing the `Preferences` dialog.

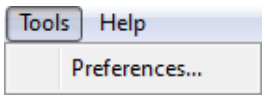

Figure 12. Tools menu

Preferences

`Preferences` opens the Preferences dialog, a full description of which can be found in Section 3 Preferences. The dialog can also be opened via a shortcut on the main toolbar.

2.2.8. Help Menu

The `Help` menu (Figure 13) is used to access version information, provide feedback, and force updates for PVRShaman.

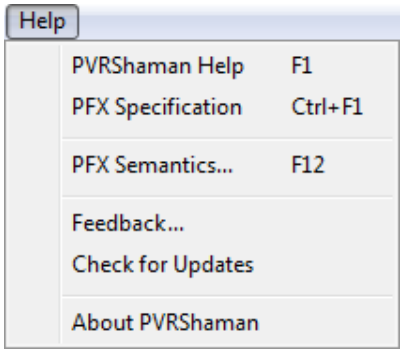

Figure 13. Help menu

View PVRShaman User Manual

To view the "PVRShaman User Manual", click `Help -> PVRShaman Help`.

View PFX Specification Document

To view the "PFX Specification", click `Help -> PFX Specification`.

View PFX Semantics

`PFX Semantics…` opens a window containing a list of all of the semantics available to PVRShaman. This list is also available in Appendix B.

Submit Feedback

To provide feedback, click `Help -> Feedback…` This will open a dialog box where instructions are displayed on how to post feedback and request for support.

Check for Updates

As of SDK release 3.0, PVRShaman is able to auto-update. However, to force-check for software updates, click `Help -> Check for Updates`.

About PVRShaman

To view basic information about PVRShaman release information such as versioning and contact details, click `Help -> About….`

2.3. Working with a Scene

2.3.1. Visualizing a Scene

The `Visualization Panel` is the area that displays the output of the combined scene data and materials, as well as any meshes in the scene with the corresponding materials applied (Figure 14). Lights and cameras are also displayed: lights as a small light bulb, and cameras as a small green camera. In instances where an object is drawn without a material it will appear entirely grey. This can be due to an error in the material that is applied or due to a lack of an applied material. More information on the cause is revealed in the `Task List` window.

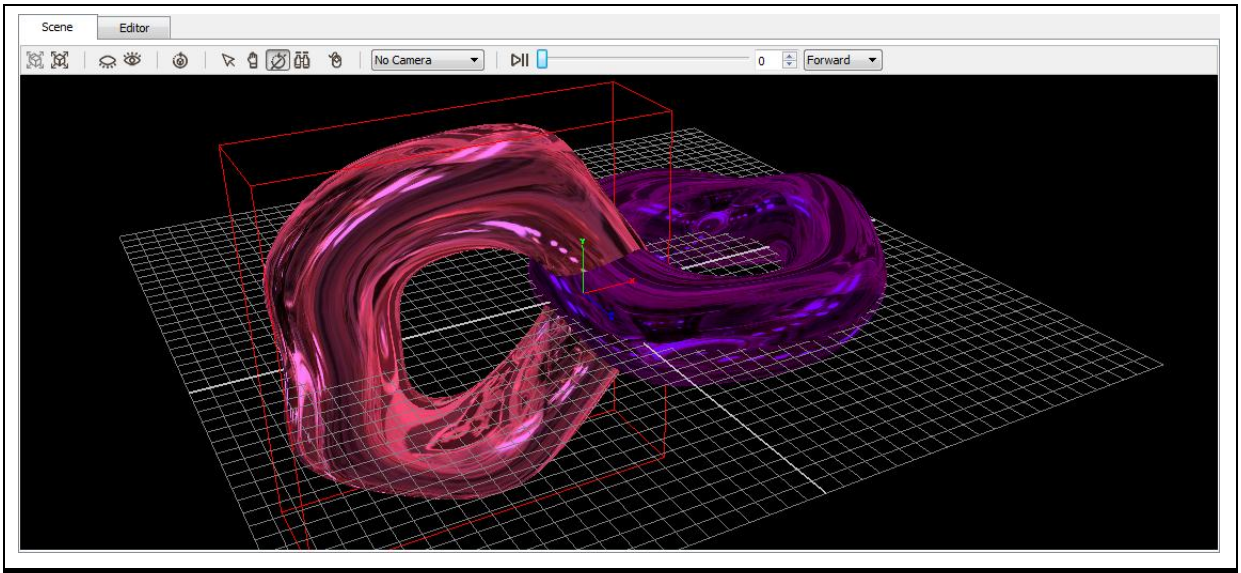

Figure 14. Visualization Panel

Clicking an object in the scene selects it. When an object is selected right-clicking will bring up a contextual menu. Finally, scenes can be explored using the mouse and keyboard.

Camera Selection

The Visualization Panel toolbar features a pull down menu that permits the selection of the current camera for the viewport. `No Camera` means that the current view is not from a camera defined in the scene. `Follow Selected` follows the currently selected object which is useful for observing one object through an animated scene.

Scene Progress

The progress bar in the toolbar shows the progress through the current scene. The box next to it shows the `Current Frame` section of the Toolbar. The number entered in here represents the frame currently being displayed.

Time Direction

Another dropdown menu is available in the toolbar, with which the direction of time within an animated scene can be selected. The three options are:

- **Forward**: The standard direction of the scene.
- **Reverse**: The reverse of the direction indicated by `Forward`.
- **Ping-Pong**: The scene will play `Forward` until completion, and then will play `Reverse` until the first frame is returned to.

2.3.2. Exploring a Scene

View Objects in a Scene

Objects in a scene are displayed in the `Scene Explorer` window (Figure 15). This also shows object meshes as well as any material applied to them. If a mesh has a parent, this will be reflected in the tree structure. An object can be selected with a single click, highlighting the model, while double clicking the object name will centre the model in the `Visualization Panel`.

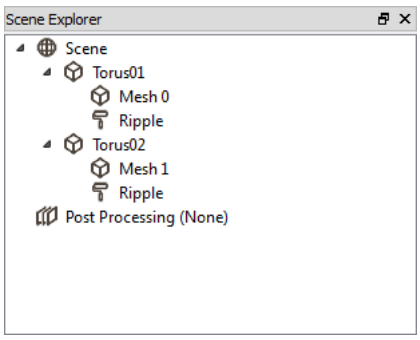

Figure 15. Scene Explorer window

Right-clicking the model name will open a context menu with options to:

- Show references, i.e. the linkage between scene nodes and POD items.
- Assign a material to the currently selected object.
- Open the currently selected object in the `View Object Data`.
- Fit the object to the screen.
- Centre the object on the screen.
- Hide the selected object.
- Hide all other objects.
- Unhide all objects.

There are no context menus on right-click for lights or cameras. However, double-clicking a camera will adjust the `Visualization Panel` to look through the associated camera.

The top portion of the `Scene Explorer` window also contains the post-processing controls. Right-clicking the currently active effect will allow the selection of the post-process (if any exists) that is currently rendered to the `Visualisation Panel`. If `None` is selected, then no post-processing is performed and the scene is rendered to the `Visualization Panel` as normal.

View Components of a Scene

The `POD Explorer` window displays the stored components of the scene, including materials, effects files, etc. (Figure 16). Expanding a component will display its properties and double-clicking a property will bring up options detailed further in Section 2.3.4.

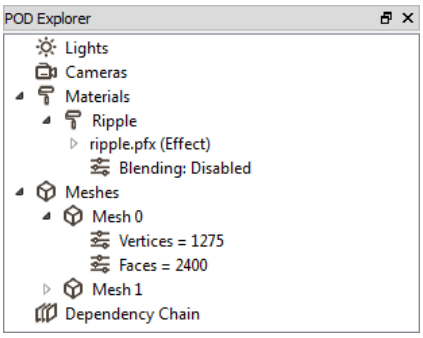

Figure 16. POD Explorer window

There are two types of materials: A single textured material, which can either be exported with a scene from Autodesk 3ds Max or Maya, or setup as described in the Add a Single Texture section; and a 'shader' material composed of a single PFX file. Double-clicking a PFX file will bring up the integrated PVRShaderEditor interface. Expanding the PFX item will show any external GLSL or texture files used within the PFX. Double-clicking a GLSL file will open it in the integrated editor. Double-clicking a texture file will open it in PVRTexTool if its location has been set in `Tools -> Preferences -> Path to PVRTexTool` or in the `Texture Viewer` if it has not.

On the other hand, `Dependency Chain` displays an ordered list of the render dependencies of the currently selected post-process effect.

View Object Data

The `Object Data` dialog (Figure 17) displays all of the data associated with a particular mesh, including vertex positions, normal directions, bone indices, vertex colours, etc. It can be accessed by right-clicking an object in the top portion of the `Scene Explorer` window and selecting `Open Data Viewer`. This is particularly useful for spotting errors in models or in the exporting process.

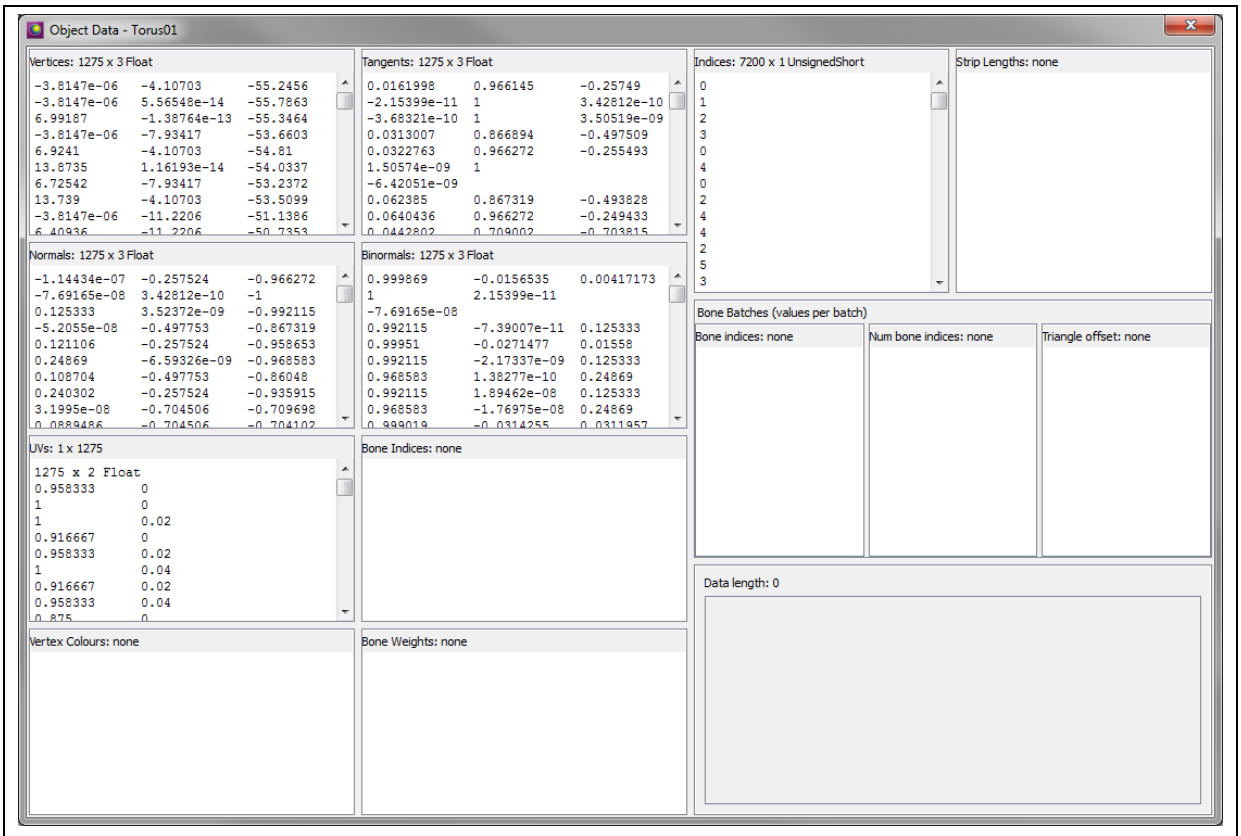

Figure 17. Object Data dialog

2.3.3. Adding Materials to a Scene

When working on scene, it may become necessary to add a new material to change the scene's appearance. Materials are added to a scene in the same way regardless of whether the scene has been imported from another program or is based on a built-in object. To add a material to PVRShaman:

1. Click `Edit Scene -> Add Material`, press `Ctrl+M` on the keyboard, or select the shortcut on the main toolbar.
2. This will open the `Add Material` dialog box as illustrated in Figure 18.
3. Give the material a name relevant to what it does (e.g., `ParallaxBumpMap`). This name will be used to attach the material to objects within the scene.

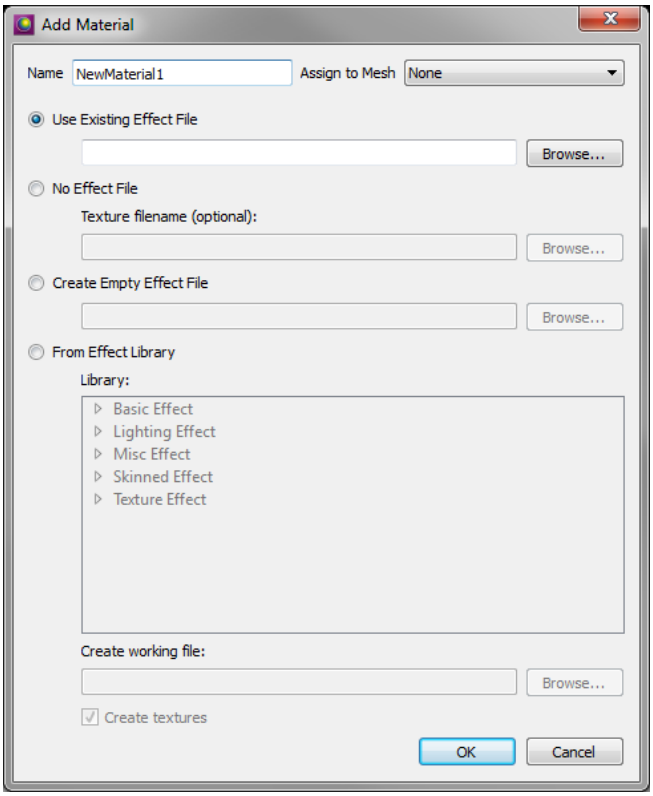

Figure 18. Add Material dialog

Add a Material Using an Existing Effects File

An existing effects file is a PFX file that contains pre-written shader code. The file may reference a number of textures in the form of PVR files. These files must be present in the same folder as the PFX file. In order to use an existing effects file:

1. Click the radio button next to `Use Existing Effects File` and click `Browse`. This will open an `open file` dialog box.
2. Browse to the location of the PFX file, and either double click it, or single click it and click `Open`. This will return the user to the `Add Material` window (Figure 18).
3. Click `Accept` to add the effects file to the scene, it will now appear under `Materials` in the `Scene Container`, the Visualization Panel will lose focus, the Effects Editor will gain it, and the shader code will now be visible.

Add a Material without an Effects File

It is possible to add a material without an effects file. To do this, click the radio button next to `No Effect File`, and then click `Accept`. The new material will now appear under `Materials` in the `Scene Explorer` window. This material can be assigned to an object where it will retain its original colour but will now be affected by any lights within the scene.

Add a Single Texture as a Material

To add a single texture, without an effects file, as a material:

1. Click the radio button next to `No Effect File`.
2. Click `Browse`. This will open an `open file` dialog box.
3. Browse to the location of the texture file then either double click it, or single click it and click `Open`.

4. Click `Accept`.

Note: The POD file format does not store the full file path to texture files. In order to avoid errors textures should be stored in the same folder as the POD file that uses them. A warning will appear when using this option on a built-in object or if the texture is not in the same folder as the POD file that is being edited or created.

Add a Material with a Blank Effects File

A blank effects file is a PFX file with some basic formatting information to help write shaders, as shown in Figure 19.

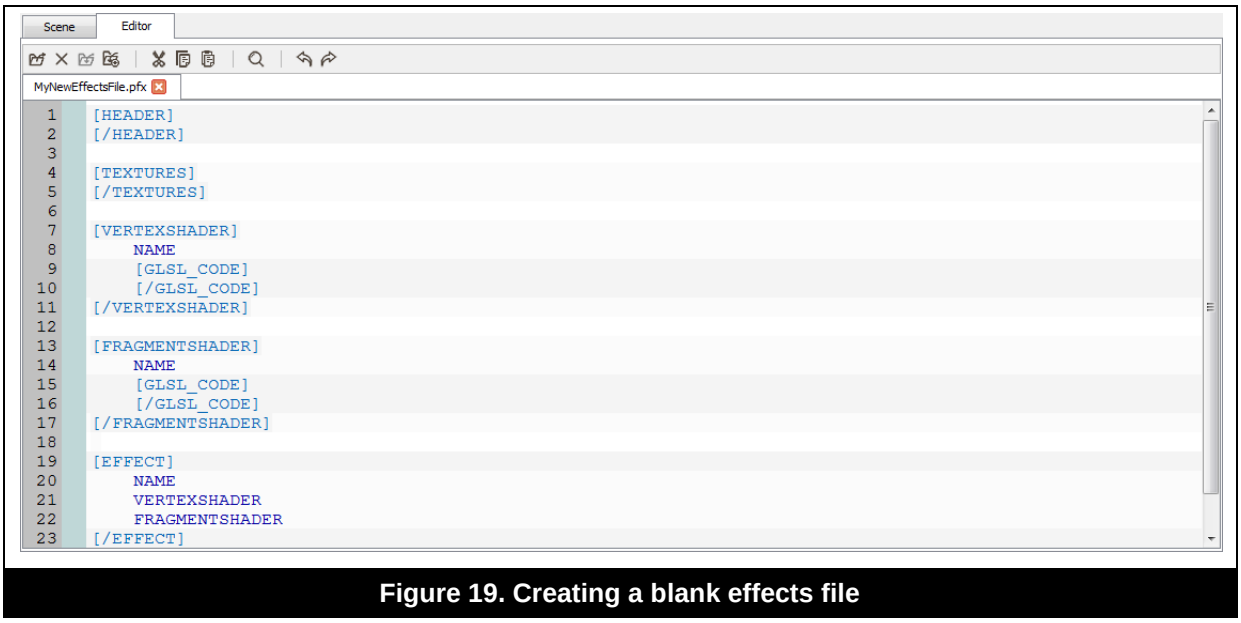

Figure 19. Creating a blank effects file

In order to add a material with a blank effects file:

1. Select `Add New Material`.
2. Click the radio button next to `Create Empty Effect File`.
3. Click `Browse`. This will open a `save file` dialog box.
4. Using this dialog box browse to a suitable location, enter a name for the file, click `Save`.
5. Click `Accept`.

Note: The POD file format does not store the full file path to texture files. In order to avoid errors textures should be stored in the same folder as the POD file that uses them. A warning will appear when using this option on a built-in object or if the texture is not in the same folder as the POD file that is being edited or created.

Existing Library Effects

PVRShaman includes a small library of built-in effects which can be automatically copied into a new PFX file, as listed in Table 2.

In order to make use of these examples:

1. From the `Add Material` window click the `From Effect Library` radio button.
2. From the library window select the desired effect and click `Browse`.
3. A `Save File` dialog box will open. Using this dialog box, browse to a suitable location and enter a name for the file.

4. Click `Save`, and then click `Accept`.

Table 2. Library effects

Effect	Explanation
`Basic Lighting`	Basic lighting functionality.
`Basic Texturing`	Basic texture functionality.
`Anisotropic Lighting`	A form of lighting used to mimic satin surfaces or brushed metal.
`Complex Lighting`	A shader for handling a number of standard lighting situations – this shader is not designed to be optimal for performance.
`Fast Transform and Lighting`	An efficient algorithm for basic transformation combined with diffuse and specular lighting calculations.
`Cell Shading`	A type of non-photorealistic rendering used to give a cartoon style effect.
`Cell Shading: Fog`	An implementation of linear, exponential and exponential squared.
`Cell Shading: Fog: Refraction`	A technique for mimicking the effect of light passing through a translucent surface like glass.
`Bumpmap`	A technique to fake complex geometry through the manipulation of reflected light.
`Fresnel Reflection`	A demonstration of reflections based on the incidence of the view vector taking into account the different refractive indices between two media.
`Iridescence`	A shader which mimics the behaviour of light reflecting off a surface covered by a translucent film with variable thickness.
`Cubemap Reflection`	A shader which reflects light hitting the model to a supplied cubemap, which is treated as if it surrounds the model.

Note: The POD file format does not store the full file path to texture files. In order to avoid errors textures should be stored in the same folder as the POD file that uses them. A warning will appear when using this option on a built-in object or if the texture is not in the same folder as the POD file that is being edited or created.

Apply a Material

Once a material has been added and appears in the `Scene Container` it can applied to a model. This can either be done as part of creating the effect, from the `Add Material` window, or by right-clicking a model within the `Scene Explorer` window, choosing `Select Material` and selecting the desired material. There is also a shortcut for this function on the main toolbar.

2.3.4. Modifying Properties

Modify Light Properties

The `Light Properties dialog` can be accessed by double-clicking a light in the `Scene Explorer` window.

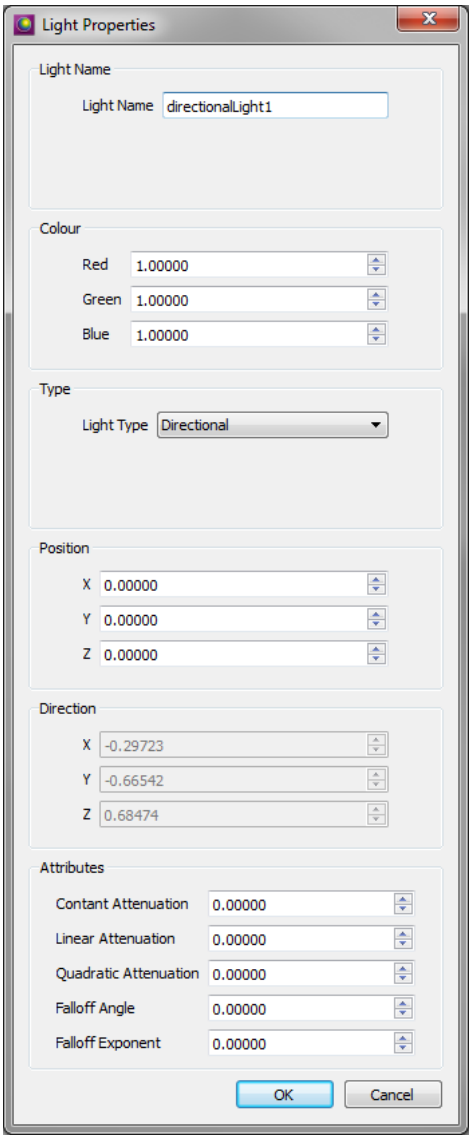

Figure 20. Light Properties dialog

It features several options, namely:

- **Colour:** `Colour` values must be in the range 0 to 1.
- **Type:** The type of light can be either `Point`, `Directional`, or `Spot`.
- **Position:** `Position` refers to the coordinates of the light source (disabled for directional lights).
- **Direction:** `Direction` refers to the direction of the light (disabled for point lights).
- **Falloff and Attenuation:** `Falloff` and `Attenuation` are for spot lights only and refer to the rate at which the light from a spot light falls off to zero based on distance and angle from the centre of the light respectively.

Changes will be displayed as they are made and saving the POD file will store the changes.

Modify Camera Properties

The `Camera Properties dialog` (Figure 21) can be accessed by double-clicking a camera in the `Scene Explorer` window.

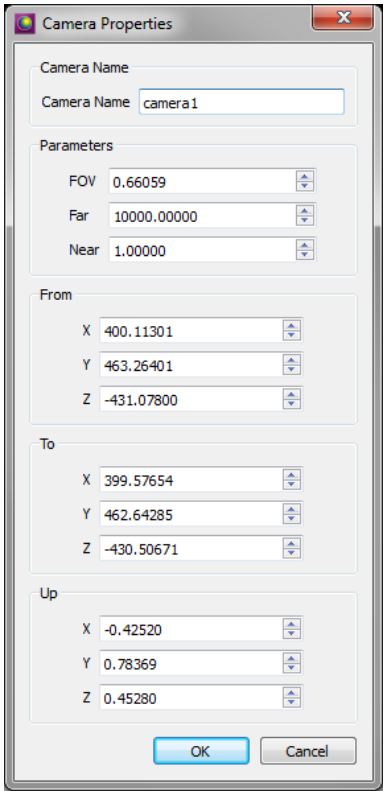

Figure 21. Camera Properties dialog

This dialog features several options, namely:

- **Parameters:** `FOV` refers to the field of view. `Near` and `Far` are the distances of the view frustum clip planes.
- **From:** `From` is the coordinate of the camera.
- **To:** `To` is the coordinate of the point the camera is looking towards.
- **Up:** `Up` refers to the direction of the up axis.

As with previous dialogs, changes are displayed as they are made.

Modify Material Properties

The `Material Properties dialog` (Figure 22) displays a material's properties and allows them to be edited. The dialog can be accessed by double clicking a material in the `Scene Explorer` window. The effects file associated with a given material can be changed as can the specific effect being used. Values for opacity, shininess, ambient colour, diffuse colour and specular colour can be adjusted in the range 0 to 1. Clicking a colour panel brings up a `Colour Selector` dialog, which serves as an alternative way to select the colour. Blend mode data can also be changed at a very precise level. All changes to the material are applied to the scene as the values are set.

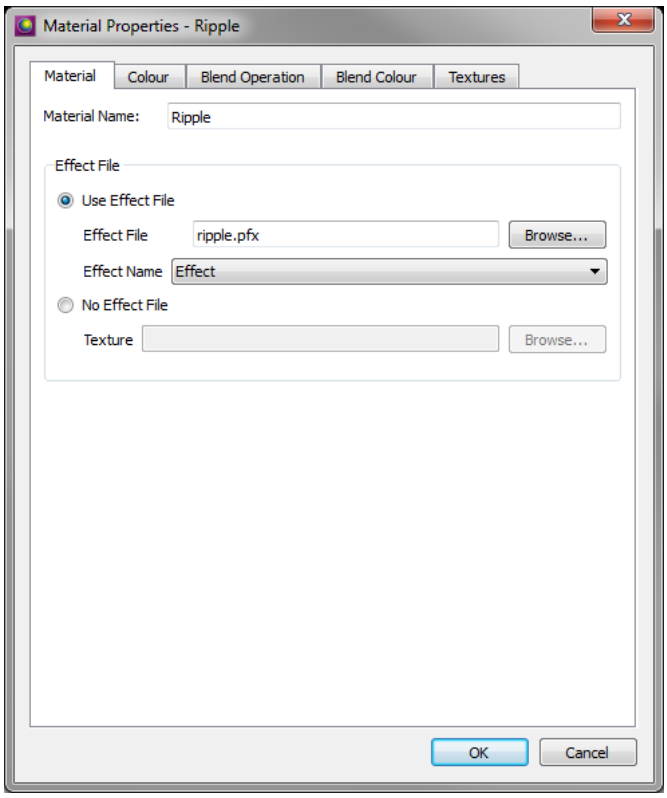

Figure 22. Material Properties dialog

2.4. Building an Effect

The PowerVR Effects (PFX) format is a small, simple, easy-to-use effects file format that allows for the declaration of 'application data' semantics so that shaders can be accurately simulated.

A single PFX file consists of several blocks that describe how a given effect is put together (see Section 2.4.1) and by default consists of a single effect. Multiple effects may exist within a single PFX file, each with their own effect block. Likewise, multiple shaders may exist within a single PFX, each referenced by name. It is also possible for separate PFX files to reference the same shaders by placing those shaders in separate files and linking to them from the vertex shader and fragment shader blocks. The full specification for the PFX format is included in the PVRShaman package and is also accessible from the PowerVR Insider website.

2.4.1. PFX Blocks

Each PFX file is broken down into several blocks, with each block serving a different purpose. A basic effect consists of a header, an effect block, a vertex shader block, a fragment shader block, and one or more texture and/or target blocks.

[HEADER]

```
[HEADER]
	VERSION		01.00.00.00
	DESCRIPTION	header example
	COPYRIGHT	Imagination Technologies
[/HEADER]
```

The header block contains information about the PFX file being edited, as explained in Table 3.

Table 3. PFX header keywords

Keyword	Explanation
VERSION	The version of the PowerVR Effects file specification that the file uses.
DESCRIPTION	A description of what the effect does.
COPYRIGHT	A copyright string.

[VERTEXSHADER] / [FRAGMENTSHADER]

```
[FRAGMENTSHADER]
	NAME		FragShader
	[GLSL_CODE]
		uniform sampler2D	sampler2d;
		varying highp vec2	texCoordinateMain;

		void main (void)
		{
			gl_FragColor = texture2D(sampler2d, texCoordinateMain);
		}
	[/GLSL_CODE]
[/FRAGMENTSHADER]

[VERTEXSHADER]
	NAME		VertShader
	FILE		VertShader.vsh
[/VERTEXSHADER]
```

The vertex shader block describes a given vertex shader. It contains up to two keywords and one nested block, listed in Table 4:

Table 4. Shader block keywords

Keyword	Explanation
NAME	The name for the vertex shader, for use in the effect block.
FILE	The FILE tag can be used to specify a file that contains the vertex shader in cases where the GLSL code block is not used.
[GLSL_CODE]	Code within the GLSL code block is passed verbatim to the GLSL compiler, and represents the actual shader code to be run.

[TEXTURE]

```
[TEXTURE]
       NAME                  Lena
       PATH                  "LenaPVR"
       MINIFICATION          LINEAR
       MAGNIFICATION         LINEAR
       MIPMAP                NEAREST
       WRAP_T                REPEAT
       WRAP_S                CLAMP
[/TEXTURE]
```

The texture block contains information about a texture to be loaded, and the flags to be used when drawing it, listed in Table 5.

Table 5. Texture block keywords

Keyword	Explanation
NAME	The name of the texture, for use in the effect block.
PATH	The path to the file the texture block represents. Currently this file must be in the same folder as the PFX file.
MINIFICATION	This option states how to handle texture minification. Two valid values exist, either LINEAR or NEAREST
MAGNIFICATION	This option states how to handle texture magnification, two valid values exist, either LINEAR or NEAREST.
MIPMAP	This option states how to pick the correct MIP map. Three valid values exist, either LINEAR, NONE, or NEAREST.
WRAP_T/WRAP_S	These two options indicate which wrap mode is to be used for the texture. Two valid values exist, either REPEAT or CLAMP.
VIEW/CAMERA	Used when performing a render-to-texture. These tags indicate which camera should be rendered to this texture. Valid values are PFX_CURRENTVIEW, the currently selected view in PVRShaman, or the name of any camera from the POD file.

When no values are provided the following values will be used:

- MINIFICATION – NEAREST
- MAGNIFICATION – NEAREST
- MIPMAP – NONE
- WRAP_T/WRAP_S – REPEAT

[TARGET]

```
[TARGET]
        NAME                    Mix
        SURFACETYPE             RGB888
        RESOLUTION              512 512
        MINIFICATION            LINEAR
        MAGNIFICATION           LINEAR
[/TARGET]
```

The target block contains information about a render target, displayed in full in Table 6. The target's name is used in an effects block to signify that the effect associated with that block should render to the named off-screen buffer.

Table 6. Target block keywords

Keyword	Explanation		
NAME	The name of the target, for use in the effect block.		
SURFACETYPE	The pixel format of the target, currently the following formats are accepted:	RGBA	8888
		RGBA	4444
		RGB	888
		RGB	565
RESOLUTION	This specifies the resolution of the target buffer in the form <X-SIZE> <Y-SIZE>, where size is a number of pixels.		
MINIFICATION	This option states how to handle texture minification. Two valid values exist, either LINEAR or NEAREST.		
MAGNIFICATION	This option states how to handle texture magnification Two valid values exist, either LINEAR or NEAREST.		

[EFFECT]

```
[EFFECT]
        NAME Bumpmapping
        VERTEXSHADER VertShader
        FRAGMENTSHADER FragShader

        TEXTURE 0 base
        TARGET  COLOR0  rendertarget

        ATTRIBUTE       inVertex                POSITION
        ATTRIBUTE       inNormal                NORMAL
        ATTRIBUTE       inTexCoord              UV
        ATTRIBUTE       inTangent               TANGENT
        UNIFORM         MVPMatrix               WORLDVIEWPROJECTION
        UNIFORM         LightPosition           LIGHTPOSMODEL0
        UNIFORM         sampler2D               TEXTURE0
 [/EFFECT]
```

The effect block is used by PVRShaman to set up the shader. It defines all the attributes and uniforms to be passed to the shader, a name for the shader and the names of the vertex and fragment shaders that are to be used for this given effect, all of which are listed in Table 7. It can also be used to set annotations, and to set the texture number for a given texture. Seven keywords, and one nested block are available for use:

Table 7. Effect block keywords

Keyword	Explanation
NAME	The name for the overall effect.
UNIFORM	Used multiple times to set-up a given uniform in the shader. This is written in the form UNIFORM <NAME> <SEMANTIC> where the name represents the name of the PVRShaman semantic list (see Appendix B).
ATTRIBUTE	Used multiple times to set-up a given attribute in the shader. This is written in the form ATTRIBUTE <NAME> <SEMANTIC> where the name represents the name of the attribute as it is written in the shader and the semantic represents a semantic from the PVRShaman semantic list (see Appendix B).
TARGET	Used multiple times. This is used to set a target number and name for a given render-to-texture target represented by a [TARGET] block. It is declared in the form TARGET COLOR<NUMBER> <NAME> where number represents the ID of the target and the name represents the name of the texture as set in the [TARGET] block. Its purpose is to specify the render target to be used by an effect.
TEXTURE	Used multiple times. This is used to set a texture number and name for a given texture set by a texture block. It is declared in the form TEXTURE <NUMBER> <NAME> where number represents the ID of the texture and name represents the name of the texture as set in the texture block.
FRAGMENTSHADER	Used to specify the fragment shader to be used for this effect. This is required as multiple fragment shaders can be specified within a single PFX file.
VERTEXSHADER	Used to specify the vertex shader to be used for this effect. This is required as multiple vertex shaders can be specified within a single PFX file.
[ANNOTATION]	All text within the annotation sub block will be read into a string. This string is not used in PVRShaman, although it can be used to pass information to other applications that may later read the PFX file.

2.4.2. Semantics

In order for a shader to function its parent application must pass certain values to it, namely uniforms and attributes. A uniform is a value that is read-only, and does not change during a render, for example light position or colour. Attributes are input values that are different for every vertex, they are also read-only, and are only available in the vertex shader.

Many of these values will change from shader to shader, and will represent radically different things. An example might be that one shader requires only the ModelViewProjection matrix, whereas another might need the inverse transpose of the ModelView matrix. Given this wide area of requirement, PVRShaman must have a means to understand what a given uniform or attribute represents so that the value can be provided to the shader when executing. This is the purpose of the PVRShaman semantics, a full list of which can be found in Appendix B.

2.4.3. PVRShaderEditor

PVRShaderEditor is the PowerVR shader and kernel editor, available as both a standalone application and as a built-in component of PVRShaman. PVRShaderEditor features the following:

- Syntax highlighting.
- On-the-fly cycle counts.
- Register information.

By default the cycle count for the currently selected file (the total of all available shaders, as well as all paths through those shaders) is under the `Profile Output Panel` of the `Compiler Output` window. Cycle counts are also available on a per-line basis allowing to quickly spot bottlenecks and see the effect of optimizations as they are carried out, as displayed in Figure 23.

Figure 23. Per-line cycle counts

It should be noted that these cycle counts are only approximations, and only available once a shader is capable of being compiled, i.e., functional. The cycle counts can be updated in one of several ways, either by clicking the `Apply Shaders` button in the toolbar or in the `Build` menu, by clicking `Rebuild Shaders` or by pressing the `F5` key.

2.4.4. Profile Output Panel

The `Profile Output Panel` is a section of the `Compiler Output` window and contains more detailed and accurate profiling information for the shader currently selected from the dropdown menu, as shown in Figure 24. This information is gained by running the shader through a cycle-accurate emulator. The information displayed in the `Profile Output Panel` is explained further in Table 8.

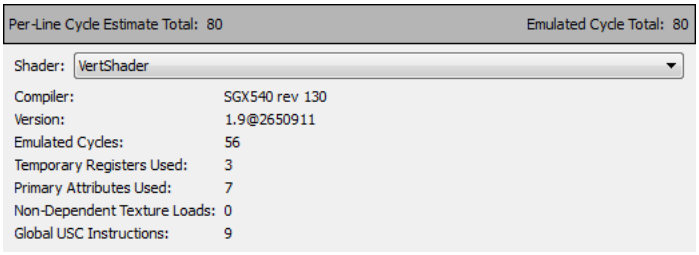

Figure 24. Profile Output Panel

Table 8. Profile Output Panel

Keyword	Explanation
`Compiler`	This value identifies the graphics core name targeted by the compiler.
`Version`	This value shows the compiler version string, where available.
`Emulated Cycles Best`	This number represents the number of cycles the shader will use when all conditional branches fail and are not processed. Emulation gives a much more accurate measuring of cycles than the per-line cycle count.
`Emulated Cycles Worst`	This number represents the number of cycles the shader will use when all conditional branches succeed and are processed. In many cases the best and worst cycle counts will be identical.
`Emulated Cycles`	This number represents the number of cycles the shader will us, and is only displayed when there are no conditional blocks present.
`Primary Attributes Used`	`Primary Attributes` are the number of logical input entities. The number of registers consumed by a primary attribute will depend on the number of elements and on its precision. For indication only; input data storage typically has 1280 registers allocated (varying from platform to platform). These registers are shared by all the shaders being run at any one time. Running out of registers will force reading and writing from external memory which might affect performance.
`Temporary Registers`	`Temporary Registers` are the extra data storage required to process a shader. As in the case of primary attributes, overflowing the storage allocated might cause degradation in performance. Temporary data storage typically has 384 registers allocated (this number might vary depending on the platform).
`Non-Dependent Texture Loads`	`Non-Dependent Texture Loads` represents the number of texture loads within the selected shader that can be pre-fetched; they are ‘independent’ of the processing of the shader.
`Global USC Instructions`	`Global USC Instructions` represents the number of instructions that can be extracted and performed once per frame rather than once per use of the shader.

2.5. Render-to-Texture and Post-Process Effects

2.5.1. Overview

The phrase “render-to-texture” is used to describe any scenario during the rendering of a scene where it is required that rendering be directed to another buffer other than the frame, specifically, to a texture. Post-processing refers to any full-screen technique performed on a scene after the scene has been fully composited and rendered. Instead of rendering 3D objects directly, the scene is first rendered to a texture. Pixel shaders are then used to apply filters to the texture before displaying it. Post-processing allows effects to be used that require awareness of the entire image such as high dynamic range, bloom, motion blur, screen space ambient occlusion, crepuscular rays, film grain simulation, depth of field and shadow mapping.

2.5.2. Render-to-Texture

Rendering to a texture is done in much the same way as rendering to the framebuffer. Vertex shader, fragment shader, effect, and texture blocks are created as usual, with the exception that the texture is given a `VIEW` entry, representing the name of the camera within the loaded POD file whose view is to be rendered into the texture. An example of this can be seen below:

```
[TEXTURE]
        NAME            renderTest
        RESOLUTION      512 512
        MINIFICATION    LINEAR
        MAGNIFICATION   LINEAR
        MIPMAP          NONE
        VIEW            "Camera02"
[/TEXTURE]

[EFFECT]
        NAME            RenderToTextureEffect

        ATTRIBUTE       inVertex                POSITION
        ATTRIBUTE       inTexCoord              UV0
        UNIFORM         sTexture                TEXTURE0
        TEXTURE         0                       renderTest
        VERTEXSHADER    VertShader
        FRAGMENTSHADER  FragShader
[/EFFECT]

[VERTEXSHADER]
        NAME            VertShader
        FILE            VertShader.vsh
[/VERTEXSHADER]

[FRAGMENTSHADER]
        NAME            FragShader
        FILE            FragShader.fsh
[/FRAGMENTSHADER]
```

2.5.3. Post-Process Effects

Post-process effects are created by chaining multiple effect blocks together. Using the output of one effect block, set using the `TARGET` flag, as an input to the next, set using the `TEXTURE` flag. An example of this, with an explanation of each stage, can be found below:

```
[TEXTURE]
        NAME                    Lena
        PATH                    "LenaPVR"
        MINIFICATION            LINEAR
        MAGNIFICATION           LINEAR
        MIPMAP                  NEAREST
[/TEXTURE]

[TARGET]
        NAME                    InputEffectOutput
        SURFACETYPE             RGB888
        RESOLUTION              512 512
        MINIFICATION            LINEAR
        MAGNIFICATION           LINEAR
[/TARGET]
[TARGET]
        NAME                    GreyscaleEffectOutput
        SURFACETYPE             RGB888
        RESOLUTION              512 512
        MINIFICATION            LINEAR
        MAGNIFICATION           LINEAR
[/TARGET]
[TARGET]
        NAME                    MixEffectOutput
        SURFACETYPE             RGB888
        RESOLUTION              512 512
        MINIFICATION            LINEAR
        MAGNIFICATION           LINEAR
[/TARGET]

[EFFECT]
        NAME            InputEffect

        ATTRIBUTE       inVertex                POSITION
        ATTRIBUTE       inTexCoord              UV0
        UNIFORM         sTexture                TEXTURE0
        TEXTURE         0                       Lena
        TARGET          COLOR0                  InputEffectOutput
        VERTEXSHADER    ScreenAlignedVS
        FRAGMENTSHADER  InputFS
[/EFFECT]
[EFFECT]
        NAME            GreyscaleEffect

        ATTRIBUTE       inVertex                POSITION
        ATTRIBUTE       inTexCoord              UV0
        UNIFORM         sTexture                TEXTURE0
        TEXTURE         0                       InputEffectOutput
        TARGET          COLOR0                  GreyscaleEffectOutput
        VERTEXSHADER    ScreenAlignedVS
        FRAGMENTSHADER  GreyScaleFS
[/EFFECT]
[EFFECT]
        NAME            MixEffect

        ATTRIBUTE       inVertex                POSITION
        ATTRIBUTE       inTexCoord              UV0
        UNIFORM         sOriginal               TEXTURE0
        UNIFORM         sGreyscale              TEXTURE1
        TEXTURE         0                       InputEffectOutput
        TEXTURE         1                       GreyscaleEffectOutput
        TARGET          COLOR0                  MixEffectOutput
        VERTEXSHADER    ScreenAlignedVS
        FRAGMENTSHADER  MixFS
[/EFFECT]
```

Step-by-Step Explanation

Assuming that `MixEffect` has been selected, the following steps will occur as illustrated in Figure 25:

1. `Lena.pvr`, a texture file, is used as an input for the `InputEffect` block.
2. The `InputEffect` block is run, outputting to `InputEffectOutput`.
3. `InputEffectOutput` is used as an input for the `GreyscaleEffect` block.
4. The `GreyscaleEffect` block is run, outputting to `GreyscaleEffectOutput`.
5. `InputEffectOutput` and `GreyscaleEffectOutput` are used as inputs for the `MixEffect` block.
6. The `MixEffect` block is run, rendering to `MixEffectOutput`.

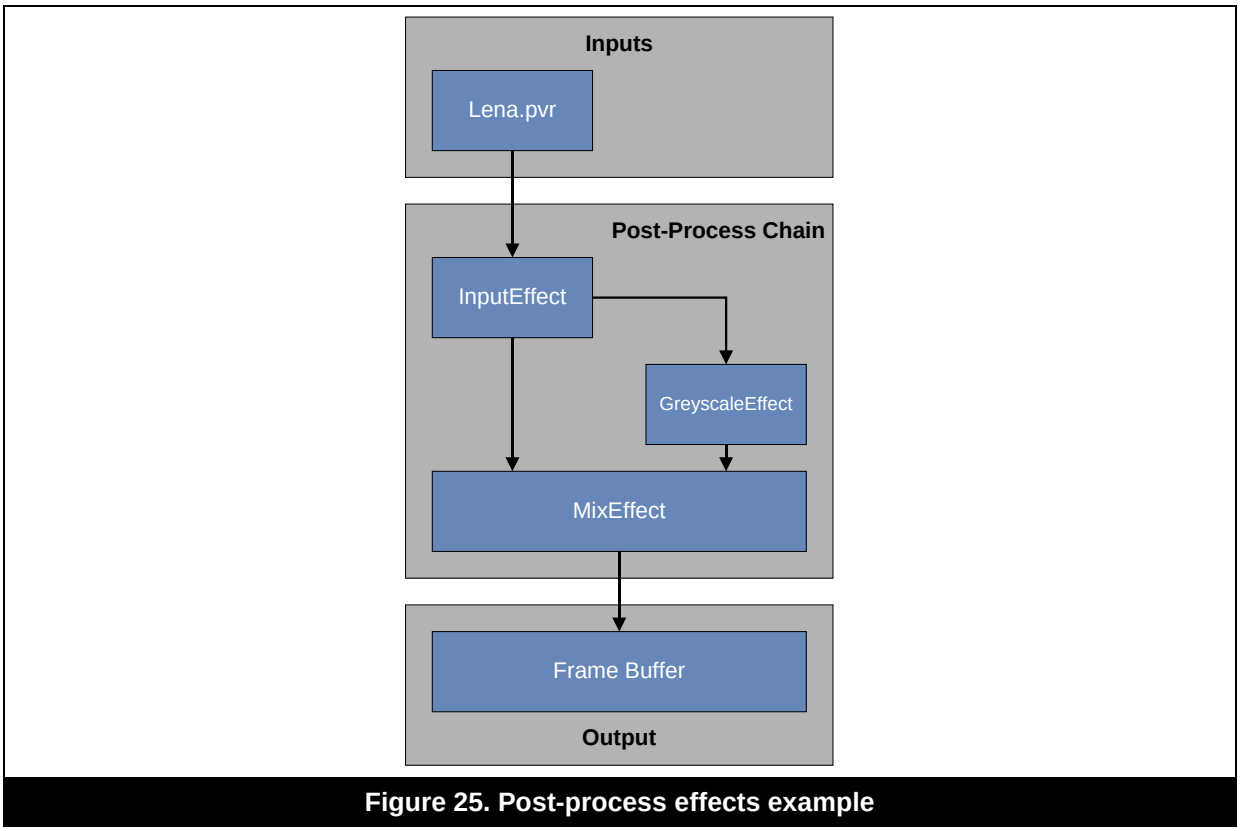

Figure 25. Post-process effects example

Outputting to the Framebuffer

When a post-process effect is selected from the PVRShaman GUI a dependency tree is built. This tree is used to determine which renders must be run and in which order to correctly display the selected effect. Once this effect has been run, its output is displayed in the `Visualization Panel`. It should be noted that the post-process effects described in this section cannot be applied to a model, unlike the effects described in Section 2.5.2.

2.6. Render Modes

These modes change the way meshes are rendered and effects are applied within the `Visualization Panel`.

2.6.1. Effects

This is the default mode, displaying any PFX and texture effects that are applied to the meshes, as illustrated in Figure 26.

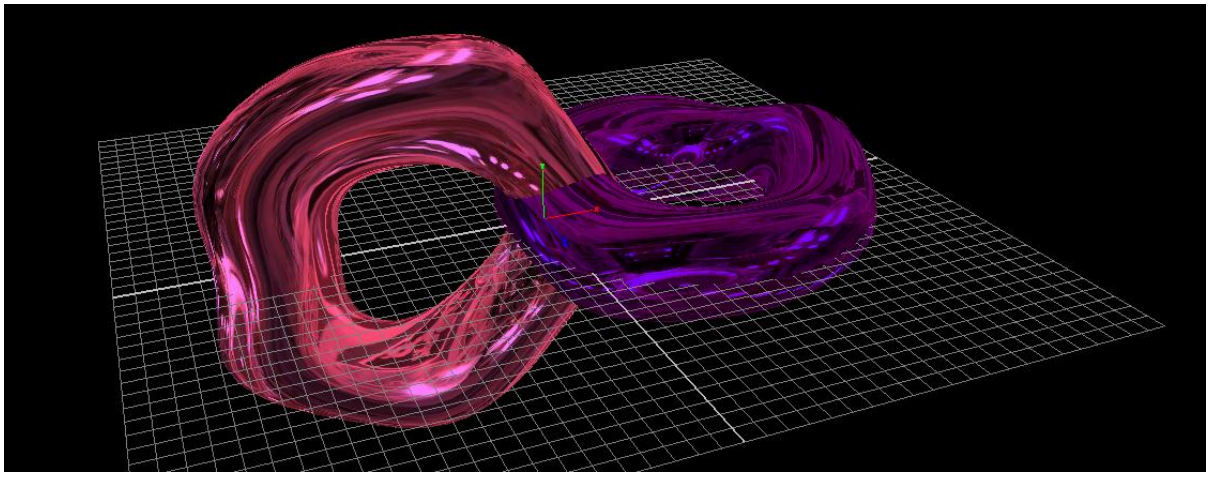

Figure 26. Effects render mode

2.6.2. No Effects

This mode shows the meshes without the effects applied, as illustrated in Figure 27. If no material is applied to a mesh, or the material fails to load, the mesh will be displayed in this way.

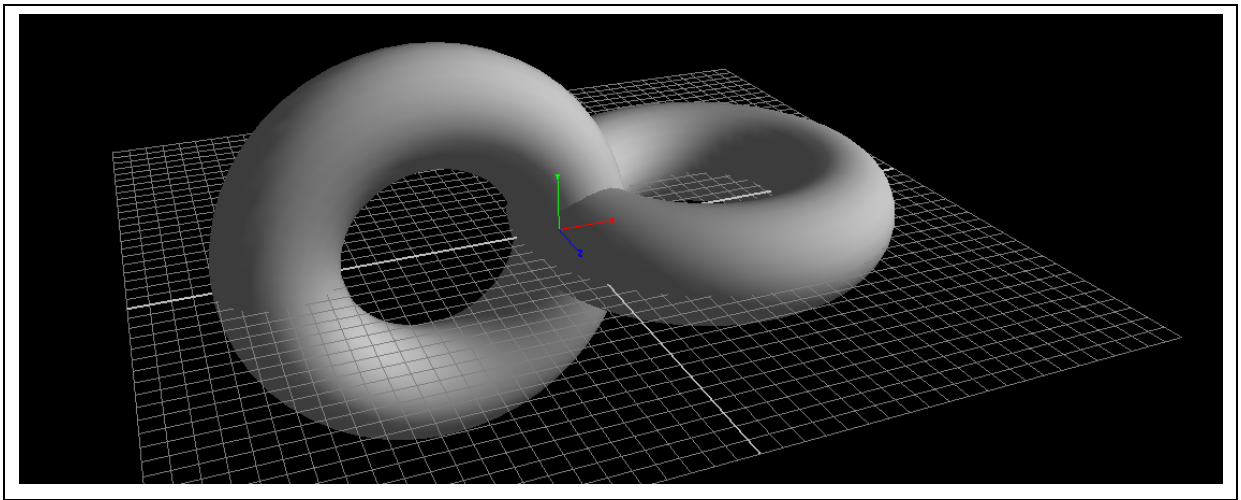

Figure 27. No Effects render mode

2.6.3. Wireframe

This mode (portrayed in Figure 28) shows the effects applied to a wireframe version of the mesh.

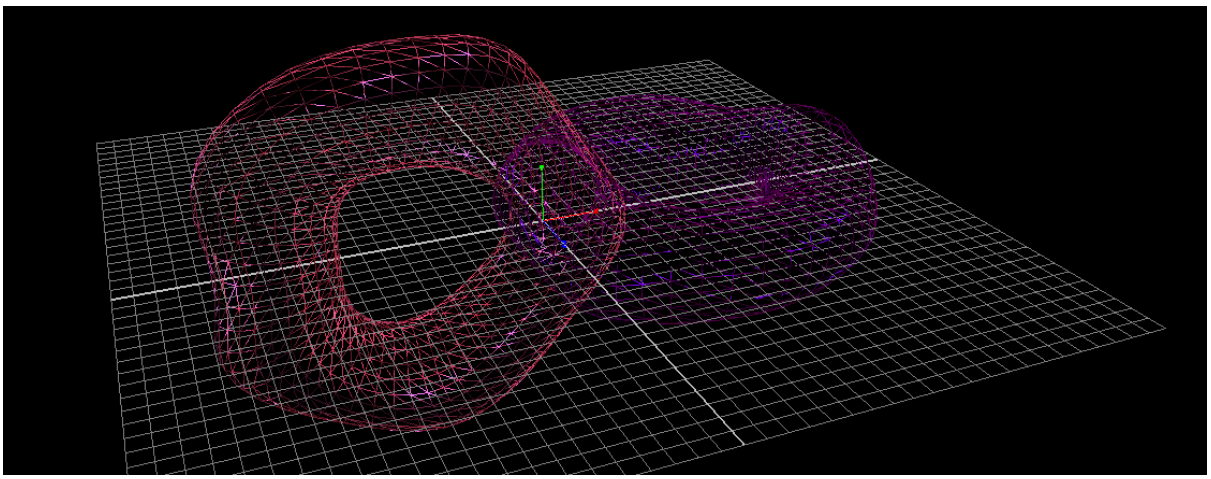

Figure 28. Wireframe render mode

2.6.4. Wireframe No Effects

This mode shows the wireframe mesh without any effects applied (as per Figure 29).

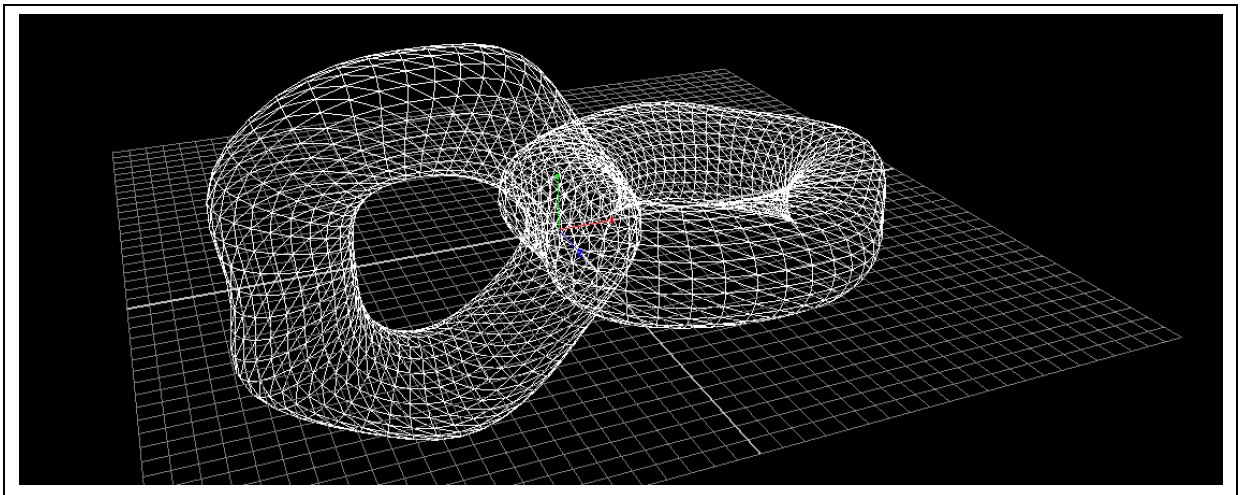

Figure 29. Wireframe No Effects render mode

2.6.5. Depth Complexity

This mode shows the depth complexity of the scene. The brighter white means greater complexity, as shown in Figure 30.

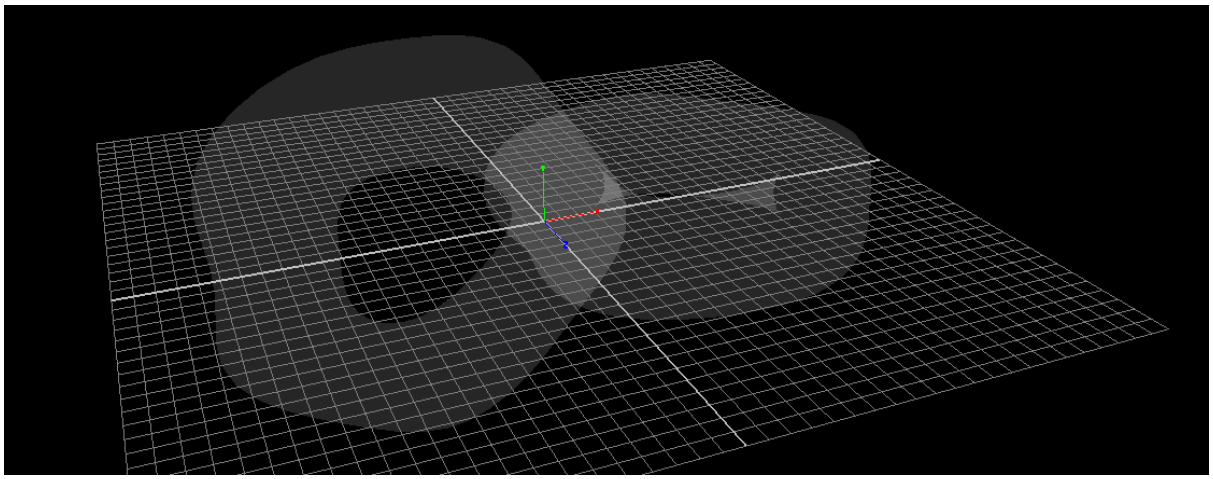

Figure 30. Depth Complexity render mode

2.7. Texture Viewer

The `Texture Viewer` displays a given texture within PVRShaman. It can be accessed by right-clicking a texture from within bottom portion of the `Scene Explorer` window. The `Texture Viewer` can also show render-to-texture targets in real-time as they update, by double-clicking the render-to-texture.

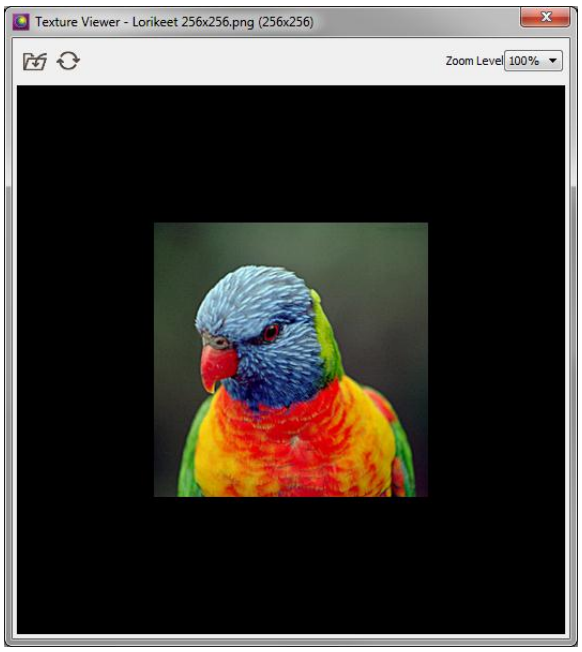

Figure 31. Texture Viewer dialog

Using the `Texture Viewer` dialog it is also possible to save a texture to a file, either as a pre-loaded texture or a given frame of a render-to-texture target. In addition, it is possible to reload a pre-loaded texture and zoom in or out on any texture.

3. Preferences

3.1. General

The `Preferences` dialog box is accessed by clicking `Tools -> Preferences…` from the menu bar. The default window displays the general options available (Figure 32).

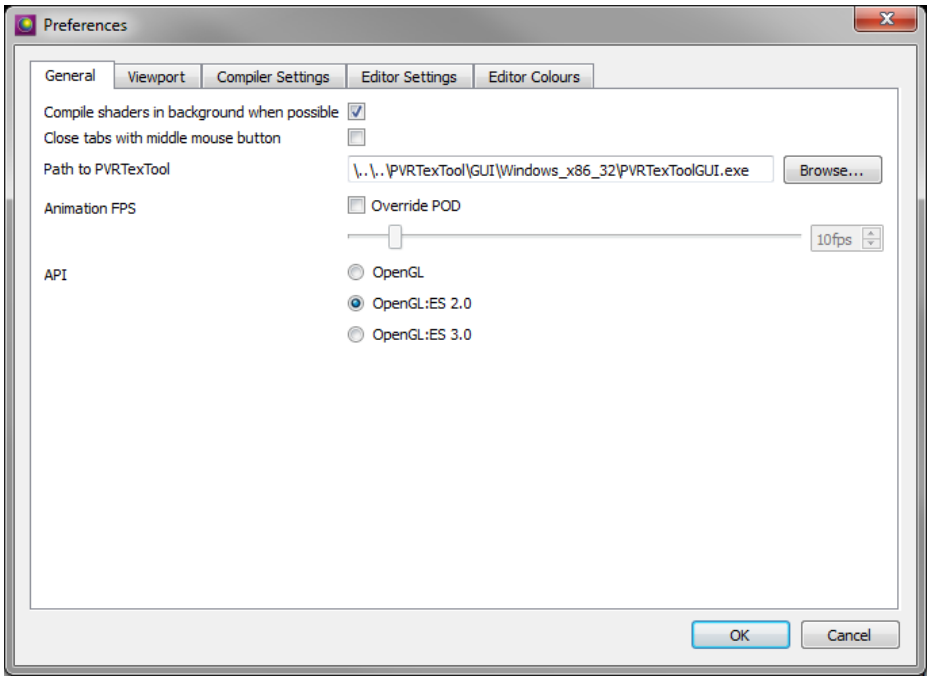

Figure 32. General tab

The options in the `General` tab are listed in Table 9.

Table 9. General options

Option	Explanation
`Compile shaders in background when possible`	With this option ticked PVRShaderEditor will attempt to compiler shaders in the background allowing it to display per-line cycle counts and profiling output while the user types.
`Close tabs with middle mouse button`	With this option ticked middle-clicking a tab in the tab bar will close the tab.
`Path to PVRTexTool`	`Path to PVRTexTool` sets the path to the PVRTexTool binary. If this is set, double-clicking a texture will open it in PVRTexTool.
`Animation FPS`	`Animation FPS` overrides the frames per second (FPS) rate of loaded POD files and will play them back at the FPS set using the slider.

3.2. Viewport

The `Viewport` tab enables the appearance of the `Visualization Panel` to be modified (Figure 33).

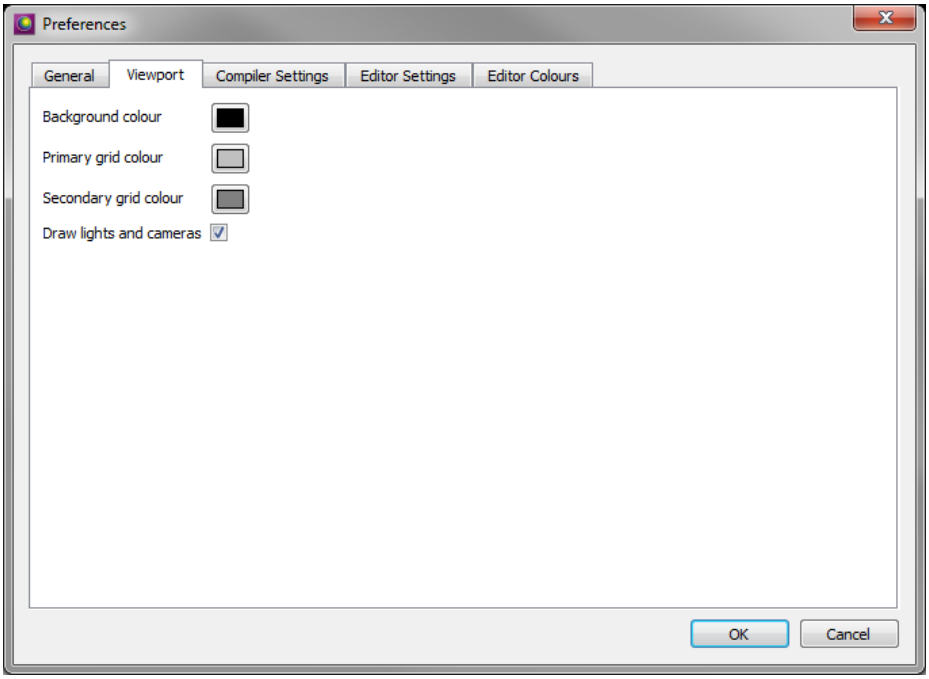

Figure 33. Viewport tab

The options in the `Viewport` tab are further explained in Table 10.

Table 10. Viewport options

Option	Explanation
`Background colour`	This option allows for the colour of the background in the `Visualization Panel` to be changed.
`Primary grid colour`	This option allows for the primary colour of the grid in the `Visualization Panel` to be changed.
`Secondary grid colour`	This option allows for the secondary colour of the grid in the `Visualization Panel` to be changed.
`Draw lights and cameras`	`Draw lights and cameras` sets whether lights and cameras have icons displayed in the `Visualization Panel`.

3.3. Compiler Settings

The `Compiler Settings` tab allows the configuration of compiler paths (Figure 34).

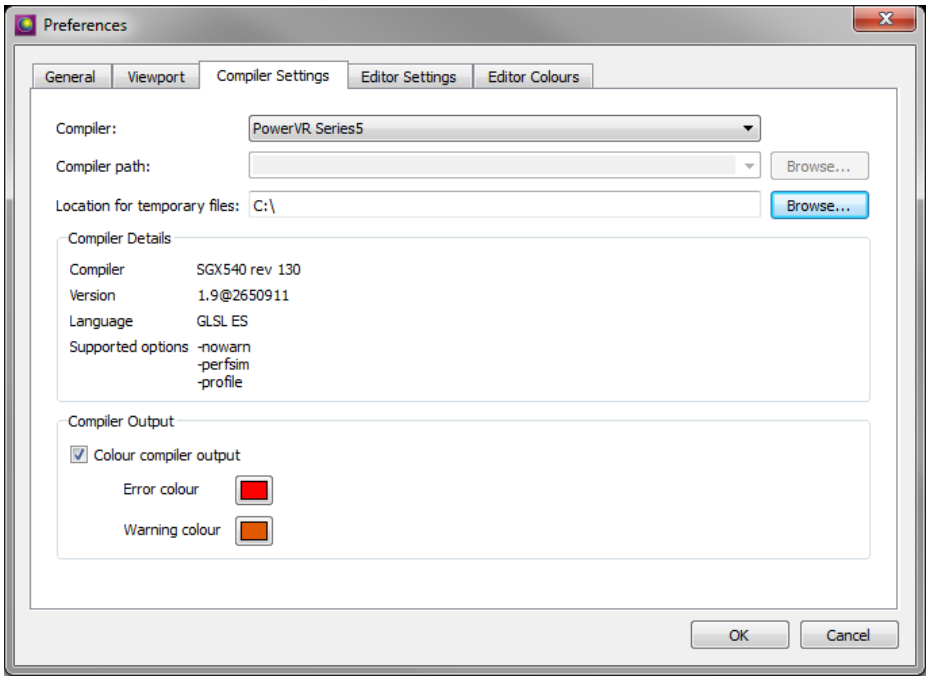

Figure 34. Compiler Settings tab

The options in the `Compiler Settings` tab are explained in Table 11.

Table 11. Compiler Settings options

Option	Explanation
`Compiler path`	`Compiler path` shows the path to the compiler currently in use.
`Location for temporary files`	This option specifies the directory the profiling information for shader will be output to for PVRShaderEditor to read.
`Colour compiler output`	With this option ticked the colouration of compiler warnings and errors can be adjusted.
`Error colour`	This button opens a colour selection window. The selected colour is used during the debugging of effects that contribute to errors.
`Warning colour`	This button opens a colour selection window. The selected colour is used during the debugging of effects that contribute to warnings.

3.4. Editor Settings

The `Editor Settings` tab allows for the changing of settings relating to the editor (Figure 35).

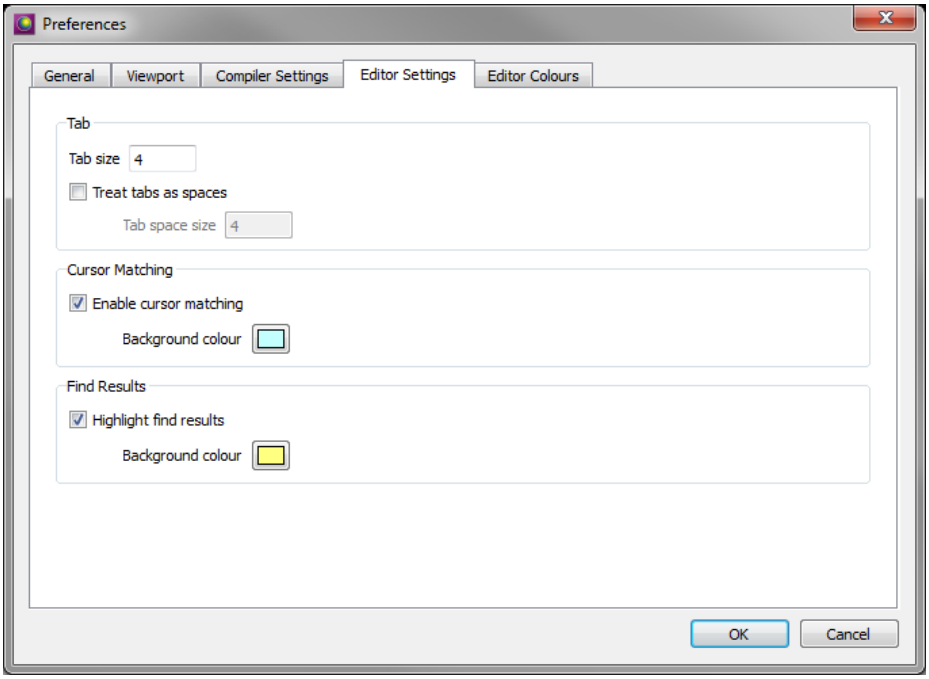

Figure 35. Editor Settings tab

The options in the `Editor Settings` tab are explained in Table 12.

Table 12. Editor Settings options

Option	Explanation
`Tab size`	`Tab Size` sets the number of spaces a tab will use if the option `Treat tabs as spaces` is ticked.
`Treat tabs as spaces`	If `Treat Tabs as Spaces` is ticked PVRShaderEditor will treat [tab] as a sequence of spaces.
`Enable cursor matching`	With this option ticked all text in the `Effects Editor` matching the text under the cursor will be highlighted.
`Background colour`	This button opens a colour selection window. The selected colour is used in the `Effects Editor` for highlighting text selected by `Enable Cursor Matching`.
`Highlight find results`	With this option ticked, text that matches the contents of the `Find` toolbar is highlighted in the `Effects Editor`.
`Background colour`	This button opens a colour selection window. The selected colour is used in the `Effects Editor` for highlighting text selected by `Highlight Find Results`.

3.5. Editor Colours

The `Editor Colours` tab allows controlling the appearance of the `Effects Editor` and includes everything from background colour to syntax highlighting colours for the various supported languages (Figure 36).

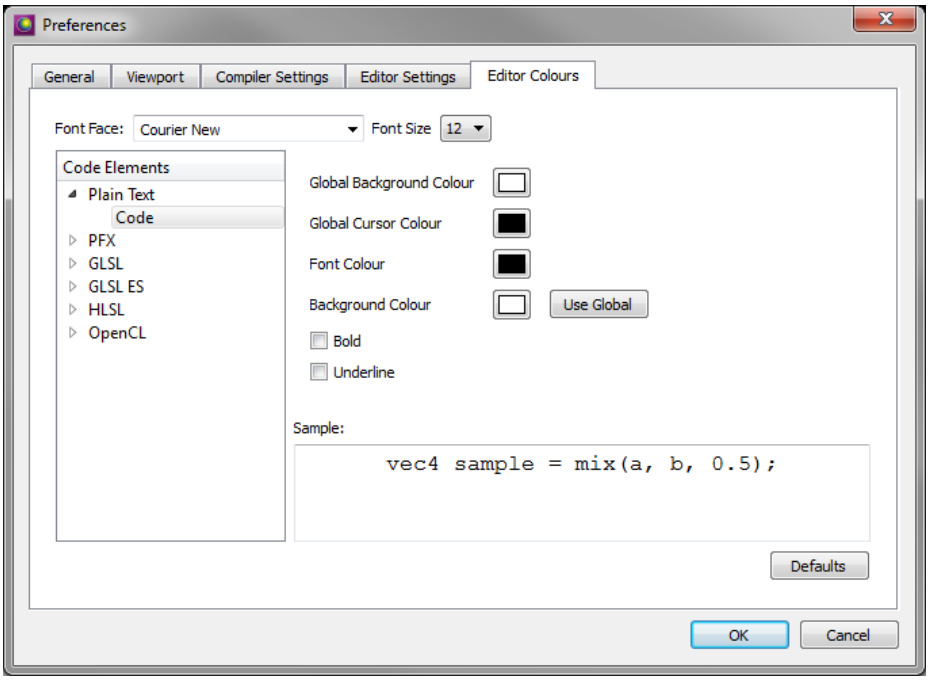

Figure 36. Editor Colours tab

The options in the `Editor Colours` tab are detailed in Table 13.

Table 13. Editor Colours options

Option	Explanation
`Font Face`	This drop down menu lists a selection of fonts available for use in the `Effects Editor`.
`Font Size`	This drop down menu allows for the selection of the size of the font used in the `Effects Editor`.
`Global Background Colour`	This option allows the colour of the background to be changed.
`Global Cursor Colour`	This option allows the cursor colour to be changed.
`Font Colour`	This option allows for the changing of the colour used in the text displayed in the `Effects Editor`.
`Background Colour`	This option allows for the selection of the colour to be used as the background of the text.
`Bold`	Ticking this box will make the text bold.
`Underline`	Ticking this box will underline the text.

4. Contact Details

For further support, visit our forum:
http://forum.imgtec.com

Or file a ticket in our support system:
https://pvrsupport.imgtec.com

To learn more about our PowerVR Graphics SDK and Insider programme, please visit:
http://www.powervrinsider.com

For general enquiries, please visit our website:
http://imgtec.com/corporate/contactus.asp

Appendix B. PVRShaman PFX Semantics List

PFX Semantics are keywords that are used within the `[EFFECT]` block to signify what a given 'uniform' or 'attribute' refers to. In a normal application this information would be calculated on the CPU during the render loop and passed to the shader, as editing the render loop to control what data is passed to the shader is not possible in PVRShaman. These keywords allow for this information to be simulated correctly.

B.1. Attributes

Table 14 lists all the attributes used in PVRShaman.

Table 14. Attribute list

Keyword	Format description
POSITION	vec4. Position.
NORMAL	vec3. Normal.
TANGENT	vec3. Tangent.
BINORMAL	vec3. Binormal.
UV[n]	vec2. n-th set of UVs. Example UV0.
VERTEXCOLOR	vec4. Vertex colour attribute.
BONEINDEX	vec4. Bone Index.
BONEWEIGHT	vec4. Bone Weight.

B.2. Uniforms

Table 15 lists the uniforms used by PVRShaman.

Table 15. Uniform list

Keyword	Format description
WORLD	mat4. World matrix.
WORLDI	mat4. World Inverse matrix.
WORLDIT	mat3. World Inverse Transpose matrix.
VIEW	mat4. View matrix.
VIEWI	mat4. View Inverse matrix.
VIEWIT	mat3. View Inverse Transpose matrix.
PROJECTION	mat4. Projection matrix.
PROJECTIONI	mat4. Projection Inverse matrix.
PROJECTIONIT	mat3. Projection Inverse Transpose matrix.
WORLDVIEW	mat4. World-View matrix.
WORLDVIEWI	mat4. World-View Inverse matrix.
WORLDVIEWIT	mat3. World-View Inverse Transpose matrix.
WORLDVIEWPROJECTION	mat4. World-View-Projection matrix.
WORLDVIEWPROJECTIONI	mat4. World-View-Projection Inverse matrix.

Keyword	Format description
WORLDVIEWPROJECTIONIT	mat3. World-View-Projection Inverse Transpose matrix.
UNPACKMATRIX	mat4. Matrix used to scale and offset vertex positions if the data has been exported with an unpack matrix.
VIEWPROJECTION	mat4. View-Projection matrix.
VIEWPROJECTIONI	mat4. View-Projection Inverse matrix.
VIEWPROJECTIONIT	mat3. View-Projection Inverse Transpose matrix.
OBJECT	mat4. Object matrix, without any parent node transformations.
OBJECTI	mat4. Object Inverse matrix, without any parent node transformations.
OBJECTIT	mat3. Object Inverse Transpose matrix, without any parent node transformations.
MATERIALOPACITY	float. Opacity of material.
MATERIALSHININESS	float. Shininess of material.
MATERIALCOLORAMBIENT	vec3. Ambient colour of material.
MATERIALCOLORDIFFUSE	vec3. Diffuse colour of material
MATERIALCOLORSPECULAR	vec3. Specular colour of material.
BONECOUNT	int. Number of bones.
BONEMATRIXARRAY	mat4[]. Array of bone transformation matrices.
BONEMATRIXARRAYIT	mat4[]. Array of bone inverse transpose transformation matrices.
LIGHTCOLOR[n]	vec3. Colour of light n (RGB). Example LIGHTCOLOR5.
LIGHTPOSMODEL[n]	vec3. Position of light n in model space. Example LIGHTPOSMODEL1.
LIGHTPOSWORLD[n]	vec3. Position of light n in world space. Example LIGHTPOSWORLD1.
LIGHTPOSEYE[n]	vec3. Position of light n in view space. Example LIGHTPOSEYE1.
LIGHTDIRMODEL[n]	vec3. Direction of light n in model space. Example LIGHTDIRMODEL1.
LIGHTDIRWORLD[n]	vec3. Direction of light n in world space. Example LIGHTDIRWORLD1.
LIGHTDIREYE[n]	vec3. Direction of light n in view space. Example LIGHTDIREYE1.
LIGHTATTENUATION[n]	vec3. Attenuation for spot lights (constant, linear, quadratic).
LIGHTFALLOFF[n]	vec2. Falloff for spot lights (angle, exponent).
EYEPOSMODEL	vec3. Eye position in model space.
EYEPOSWORLD	vec3. Eye position in world space.
TEXTURE[n]	sampler2D. Sampler for texture n. Example TEXTURE2.
ANIMATION	float. Contains the objects distance through its animation. Range 0 to 1.
GEOMETRYCOUNTER	int. Resets to 0 at the beginning of each render frame and increases by one for each submission of geometry.
VIEWPORTPIXELSIZE	vec2. Size of the viewport in pixels.

Keyword	Format description
VIEWPORTCLIPPING	vec4. Near distance, far distance, width angle (radians), height angle (radians).
TIME	float. The current time in seconds.
TIMECOS	float. Cosine of the current time in seconds.
TIMESIN	float. Sine of the current time in seconds.
TIMETAN	float. Sine of the current time in seconds.
TIME2PI	float. Tangent of the current time in seconds.
TIME2PICOS	float. Cosine of the current time in seconds wrapped to 2*pi.
TIME2PISIN	float. Sine of the current time in seconds wrapped to 2*pi.
TIME2PITAN	Float. Tangent of the current time in seconds wrapped to 2*pi
LASTTIME	float. The last frame's time.
ELAPSEDTIME	float. The time between adjacent frames.
BOUNDINGCENTER	vec3. Bounding box centre.
BOUNDINGSPHERERADIUS	float. Bounding sphere radius.
BOUNDINGBOXSIZE	vec3. Bounding box size.
BOUNDINGBOXMIN	vec3. Bounding minimum for x, y, z.
BOUNDINGBOXMAX	vec3. Bounding maximum for x, y, z.
RANDOM	float. A random value (Range 0 to 1).
MOUSEPOSITION	vec3. The mouse position on screen (x, y, time).
LEFTMOUSEDOWN	vec4. The left mouse down state, and its position at that time (x, y, isdown, timedown).
RIGHTMOUSEDOWN	vec4. The right mouse down state, and its position at that time (x, y, isdown, timedown).

Appendix C. Regular Expression Syntax

Table 16 lists the regular expression syntax used by PVRShaman.

Table 16. Regular expression syntax and description

Special constructs	
`(?i X )`	Match sub pattern, case insensitive
`(?I X )`	Match sub pattern, case sensitive
`(?n X )`	Match sub pattern with newlines
`(?N X )`	Match sub pattern with no newlines
`( X )`	Capturing parentheses (use with back references)
`(?: X )`	Non-capturing parentheses
`(?= X )`	Zero width positive look ahead
`(?! X )`	Zero width negative look ahead
`(?<= X )`	Zero width positive look behind
`(?<! X )`	Zero width negative look behind
`(?> X )`	Atomic grouping (possessive match)
Logical operators	
`X Y`	X followed by Y
`X | Y`	Either X or Y
Quantifiers	
`X *`	Match 0 or more
`X +`	Match 1 or more
`X ?`	Match 0 or 1
`X {}`	Match 0 or more
`X {n}`	Match n times
`X {,m}`	Match no more than m times
`X {n,}`	Match n or more
`X {n,m}`	Match at least n but no more than m times
Note: These quantifiers are greedy. By following them with `?` it becomes possible to turn them into lazy quantifiers, or follow them with `+` for possessive (non-backtracking) quantifiers.	
Boundary matching	
`^`	Match begin of line [if at begin of pattern]
`$`	Match end of line [if at end of pattern]
`\<`	Begin of word
`\>`	End of word
`\b`	Word boundary
`\B`	Word interior
`\A`	Match only beginning of file

\Z	Match only end of file
Character classes	
[abc]	Match a, b, or c
[^abc]	Match any but a, b, or c
[a-zA-Z]	Match upper or lower-case a through z
[]]	Matches]
[-]	Matches -
Predefined character classes	
.	Match any character
\d	Digit [0-9]
\D	Non-digit
\s	Space
\S	Non-space
\w	Word character [a-zA-Z_0-9]
\W	Non-word character
\l	Letter [a-zA-Z]
\L	Non-letter
\h	Hex digit [0-9a-fA-F]
\H	Non-hex digit
\u	Single uppercase character
\U	Single lowercase character
\p	Punctuation (not including '_')
\P	Non punctuation
Characters	
\\	Back slash character
\033	Octal
\x1b	Hex
\t	Tab
\n	Newline
Back references	
\1 to \9	Reference to 1st to 9th capturing group